National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

University of Alberta

An Adaptive Controller for a Fast Moving Six-joint Robot Manipulator

©

by

Won-Kuk Son

A thesis
submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree
of Master of Science

Department of Electrical Engineering
Edmonton, Alberta
Fall 1993

National Library of Canada

Acquisitions and Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada

Direction des acquisitions et des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-88144-5

Canada

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: Won-Kuk Son
TITLE OF THESIS: An Adaptive Controller for a Fast Moving Six-joint Robot Manipulator

DEGREE: Master of Science
YEAR THIS DEGREE GRANTED: 1993

Permission is hereby granted to the UNIVERSITY OF ALBERTA LIBRARY to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication rights and other rights in association with the copyright of the thesis, and except as hereinbefore provided neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

(Signed) ____________
Permanent Address:
Department of Electronic Engineering,
Sung Kyun Kwan University, Seoul,
Korea

Date: July 8 1993

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **An Adaptive Controller for a Fast Moving Six-joint Robot Manipulator** submitted by **Won-Kuk Son** in partial fulfillment of the requirements for the degree of Master of Science.

V. Gourishankar

Dr. V. G. Gourishankar (Supervisor)

Dr. G. S. Christensen

Rao

Dr. M. Rao

Date: July 6 1993

In Gratitude To My Parents

Abstract

For the control of a revolute six-joint robot manipulator, an adaptive self-tuning controller is proposed. For simulating the motion of six-joint PUMA robot manipulator, the dynamics equation of all the six joints based on the Lagrange-Euler approach is shown in a complete closed form. This form of the equation is helpful to provide the necessary insight about the characteristics of robot dynamics for controller design.

Simulation studies, when the robot is moving at a moderately high and very high velocity with and without payload, show that the controller performance is satisfactory. The novel features in this thesis are : *a*) full revolute six-joint robot dynamics is used for the controller. *b*) a general multi-input-multi-output self-tuning controller (MIMO STC) is converted to a single-input-multi-output self-tuning controller (SIMO STC) and extended to work well at a very high velocity with a payload.

Acknowledgements

I would like to express my deepest gratitude my supervisor, Dr. V. G. Gourishankar, for his encouragement and guidance as well as patience throughout this work. I would also like to thank Dr. H. Zhang for his suggestions and advice about the project. I extend my thanks to Mr. Darin Ingimarson and Dr. X. Shen in the Robotics Laboratory for their sincere advice and help. The financial support by the Electrical Engineering Department, University of Alberta, and the Natural Sciences and Engineering Research Council of Canada (NSERC) are gratefully acknowledged, for without their support this work would not have been possible.

I would be grateful to my special friend, Heon-Chang Kim, who shared his feelings with me, and provided the necessary encouragement by sharing my experiences in times of difficulties. Finally, I would like to express my gratitude to my parents for being there for me and supporting me in all aspects of my life.

Contents

1 Introduction **1**

1.1 Need for Adaptive Control 2

1.2 Literature Survey 3

1.3 Summary 6

1.4 Objectives of Thesis 7

1.5 Organization of Thesis 8

2 Robot Systems - Background **9**

2.1 Description of Manipulator by D-H Notation 9

2.1.1 Determination of Coordinate System in Robotics . . . 11

2.1.2 Determination of the Kinematic Parameters 11

2.2 Determination of Transformation Matrix 13

2.2.1 Recursive Transformation Matrix 13

2.2.2 Partition and Decomposition of Transformation Matrix 14

2.3 Orientation of End-Effector 15

2.3.1 An Algorithm for Calculating Ψ_x, Ψ_y, and Ψ_z 17

2.4 Summary - Need for Six-joint Dynamics 20

3 Modeling of Physical PUMA Manipulator : Dynamics Model **22**

3.1 Lagrange-Euler(L-E) Dynamics Model 22

3.2 Derivation of L-E Dynamics Model for a Motion of Robot Manipulator . . . 24
  3.2.1 Lagrange-Euler Equation . . . 24
  3.2.2 Kinetic Energy of a Robot Manipulator . . . 25
  3.2.3 Potential Energy of a Robot Manipulator . . . 31
  3.2.4 L-E Dynamics Equations of a Robot Manipulator . . . 31
  3.2.5 Computational Simplification of Dynamics Equation using $Q - matrix$ . . . 35
3.3 Customized Closed-form of Dynamics Equation for Six d-o-f PUMA Robot . . . 38
  3.3.1 Customizing Dynamics Equation . . . 39
  3.3.2 Parameters of PUMA 600 . . . 41
3.4 Conclusion . . . 44

4 Adaptive Self-tuning Controller **45**
4.1 Time-Series Difference Model and Adaptive Self-tuning Controller 45
4.2 SISO Adaptive Self-tuning Controller for Independent Joint Dynamics . . . 47
  4.2.1 SISO ARX-model for Manipulator Dynamics . . . 48
  4.2.2 Parameter Estimation in SISO ARX-model . . . 49
  4.2.3 Optimal LQG Controller Design for SISO system . . . 50
4.3 MIMO Self-tuning Controller for Interacting Joint Dynamics . 52
  4.3.1 MIMO ARX-model for Manipulator Dynamics . . . 53
  4.3.2 Parameter Estimation in MIMO ARX-model . . . 53
  4.3.3 Optimal LQG Controller Design for MIMO System . . 56
4.4 MIMO Self-tuning LQG Controller for Controlling Velocity Variables . . . 57

5 Computer Simulations and Development **61**

5.1 Digital Simulation of manipulator Motion 63

5.1.1 Measured Angular Velocity and Position with Measurement Error 63

5.1.2 About Generation of Gaussian Noise 64

5.2 Design of Adaptive STC on SIMO ARX-model 65

5.2.1 Transformation MIMO ARX-model to SIMO ARX-model 65

5.2.2 Parameter Estimation in SIMO ARX-model By RLS . 67

5.3 Simulation Results on Adaptive SIMO STC 71

5.3.1 Tracking High Velocity and Position without Payload . 71

5.3.2 Tracking Very High Velocity and Position with Picking up Payload 79

5.4 Pseudo-Linearization by Weakening Effects of Couplings and Non-linearities 81

6 Conclusions and Future Work **90**

6.1 Conclusions 90

6.2 Suggestions for Future Work 91

Bibliography **93**

7 Customized L-E Forward Dynamics Equation for Six-joint PUMA 600 **98**

List of Figures

1 PUMA Manipulator with D-H Notation 10
2 Illustration of Structural Parameters 12
3 Euler Angles for Orientation 16
4 A Point $\mathbf{r}_i^i$ in Link i. 26
5 Explicit Self-tuning Control Scheme 46
6 Test1:Velocity Tracking in Joint1 without payload 73
7 Test1:Position Tracking in Joint1 without payload 73
8 Test1:Position Tracking Error in Joint1 without payload . . . 73
9 Test1:Velocity Tracking in Joint2 without payload 74
10 Test1:Position Tracking in Joint2 without payload 74
11 Test1:Position Tracking Error in Joint2 without payload . . . 74
12 Test1:Velocity Tracking in Joint3 without payload 75
13 Test1:Position Tracking in Joint3 without payload 75
14 Test1:Position Tracking Error in Joint3 without payload . . . 75
15 Test1:Velocity Tracking in Joint4 without payload 76
16 Test1:Position Tracking in Joint4 without payload 76
17 Test1:Position Tracking Error in Joint4 without payload . . . 76
18 Test1:Velocity Tracking in Joint5 without payload 77

19 Test1:Position Tracking in Joint5 without payload 77

20 Test1:Position Tracking Error in Joint5 without payload . . . 77

21 Test1:Velocity Tracking in Joint6 without payload 78

22 Test1:Position Tracking in Joint6 without payload 78

23 Test1:Position Tracking Error in Joint6 without payload . . . 78

24 Test2:Velocity Tracking in Joint1 with payload 80

25 Test2:Position Tracking in Joint1 with payload 80

26 Test2:Position Tracking Error in Joint1 with payload 80

27 Adaptive STC with Weakened Couplings and Non-linearities . 82

28 Test3:Velocity Tracking in Joint1 with payload 84

29 Test3:Position Tracking in Joint1 with payload 84

30 Test3:Position Tracking Error in Joint1 with payload 84

31 Test3:Velocity Tracking in Joint2 with payload 85

32 Test3:Position Tracking in Joint2 with payload 85

33 Test3:Position Tracking Error in Joint2 with payload 85

34 Test3:Velocity Tracking in Joint3 with payload 86

35 Test3:Position Tracking in Joint3 with payload 86

36 Test3:Position Tracking Error in Joint3 with payload 86

37 Test3:Velocity Tracking in Joint4 with payload 87

38 Test3:Position Tracking in Joint4 with payload 87

39 Test3:Position Tracking Error in Joint4 with payload 87

40 Test3:Velocity Tracking in Joint5 with payload 88

41 Test3:Position Tracking in Joint5 with payload 88

42 Test3:Position Tracking Error in Joint5 with payload 88

43 Test3:Velocity Tracking in Joint6 with payload 89

44 Test3:Position Tracking in Joint6 with payload 89

45 Test3:Position Tracking Error in Joint6 with payload 89

List of Tables

2.1 Structural Kinematic Parameters 13

3.2 Kinematic Link Parameters for PUMA 600 41
3.3 Center-Of-Mass and Relative Link Mass 42
3.4 Radii-of-Gyration 43

Chapter 1

Introduction

In order that a robot manipulator can grasp an object, the position and orientation of an object should coincide with those of the robot end-effector in three dimensions. A controller for a robot manipulator has therefore the responsibility to force the center-point and orientation of the end-effector to follow the desired trajectory of object which can be expressed as joint angles or $X - Y - Z$ coordinates(position) and Euler-angles(orientation) in three dimensional cartesian space.

As discussed in Chapter 2, a robot manipulator needs to have a minimum six degree-of-freedom(d-o-f) for an end-effector to be able to move to a specified position and orientation in three dimension workspace. The PUMA 600 is one such industrial manipulator. For simulation studies, therefore, the dynamics equation of all the revolute six joints on the Lagrange-Euler approach is used.

The control of the robot motion consists of two phases. The first is called *gross motion* control, in which controller is constructed for the actuators that make the end-effector move from initial position/orientation to the desired target position/orientation along a planned path. The second is referred to as the *fine motion* control where sensory feedback information of object for desired trajectory is dynamically fed into the controller at each or every several sampling period. In contrast to gross motion control, fine motion controller is subject to not only position/orientation but also velocity of moving object. Therefore, fine motion control should take into account for both velocity and position simultaneously. Stability, speed, and accuracy are important considerations in fine motion control for a high level of mobility and dexterity.

1.1 Need for Adaptive Control

In current industrial approaches to robot manipulator control system design, a simple joint servo mechanism is employed. This servo mechanism approach is improperly modeling the varying dynamics of robot manipulator because it neglects the motion dynamics and configuration of the whole manipulator mechanism. The conventional feedback control strategies may omit significant dynamical effects in parameters of the controlled system. For example, the PUMA robot manipulator is equipped with conventional PID controller for joint variable control. In fact, with PID control, PUMA manipulator moves with noticeable vibration, reduced speed and limited precision.

One of the main disadvantages of this control scheme is that the feedback gains are constant and prespecified. It does not have the capability of updating the feedback gains under varying payload. Without even picking up or changing payload, the dynamics of manipulator is changing rapidly depending on the desired trajectory since dynamics of manipulators is a non-linear function of structural kinematic parameters, joint positions, joint velocities and joint accelerations of manipulator. The real dynamics of manipulator is highly coupled and non-linear. At high velocities, Coriolis and centrifugal forces are introduced and must be taken into account. Therefore designing the controller under assumption that the manipulator is moving slowly, as is done in many control schemes, should be avoided for a dextrous robot arm. The inertial load at each joint varies significantly with the position of the manipulator. Parametric uncertainties may come from imprecise knowledge of the manipulator mass properties, unknown payload, uncertainty in the payload position of the end-effector. In order to handle the uncertainties and variations in dynamics of plant, that is, the varying degree of non-linearities, inertial loading, the coupling between joints and gravity effects due to above reasons, adaptive control laws are desirable to be used in compensating for these undesirable effects. The adaptive controller, compared to non-adaptive control scheme, has two additional functions.

1. **Plant Identification.** The identifier determines the dynamic characteristics of plant using measurements of input and output signals of plant.
2. **Controller Gain adjustment.** The controller gain is continually adjusted according to the measured variation in plant dynamics.

1.2 Literature Survey

Many researchers have developed adaptive control laws to compensate for effects of parameter uncertainties.

[Dubowsky&D. 79] proposed a simple model reference adaptive control for control of mechanical manipulators. They took into account the effect of payload by combining it with the final link. A Model-Reference-Adaptive Control(MRAC) law was devised using the steepest descent method for a manipulator with counterbalance in order to handle non-linearities and payload. But couplings between joints of the manipulator was neglected, and a numerical simulation study was performed with three-joint dynamics.

[Takegaki&A. 80] developed a MRAC law which consists of feedforward control and feedback control. The feedforward control reduces the effects of gravity, while feedback control compensates for the position errors, velocity errors, constant disturbances, and acceleration requirements based on the Lyapunov direct method. But they assumed low-speed motion to make it possible to neglect Coriolis and centrifugal force, and their simulation was shown with four-joint manipulator.

[Koivo&G. 83] proposed an adaptive self-tuning controller based on discrete linear time-invariant decoupled model. The controller algorithm assumes that the interaction forces among the joints are negligible. Thus, the assumption of slowly varying parameters is unavoidable. Tracking results at a high velocity are not shown. Robustness of proposed algorithm is questionable for different desired trajectories because the non-linearity in the dynamics of the manipulator strongly depends on the desired trajectories of high velocity profile.

[Lee&C. 84], [Lee&C. 85] suggested an adaptive perturbation control scheme composed of a nominal control and a variational control. Since nominal control uses the direct calculation of manipulator dynamics along the desired trajectory, it requires full information for the dynamics of manipulator. The variational control regulating the perturbation with respect to desired trajectory was based on a linear perturbation model of the manipulator together with a recursive least-squares identification algorithm and a one-step-ahead optimal control algorithm. They showed by computer simulation that their control law was insensitive to variation of payload, but convergence of the control law was not shown explicitly. Their simulation was tested with three-joint manipulator.

[Craig&H.&S. 86] presented an adaptive computed torque or inverse dynamics method for the control of manipulators with rigid links. They try to show a globally stable control scheme and conditions for parameter convergence as well as its asymptotic properties. However, lots of drawbacks exist in practice due to the fact that the global stability of this method depends on having exact dynamic models and full knowledge of parameters of the system, accuracy, and speed in computation. Thus, degradation of response may occur due to disturbance, change in payload, sensor measurement, etc.. Their simulation showed the results of link mass estimation and of one joint position error tested on dynamics of only two-joint planer manipulator. The more complete simulation results are required to show the robustness with respect to various desired paths of high velocity profiles which indeed affect the dynamics of manipulator and robustness in change or existence of payload.

[Khorrami&O. 88] proposed decentralized control of robot manipulator using state and PI feedback. This paper seems to suggest that a static state feedback is indeed sufficient to stabilize the system about a constant setpoint(desired path), and system may still perform satisfactorily if the constant reference signal is replaced by a slowly time-varying signal. However, with the assumption of constant setpoint or slowly time-varying reference input, stability or robustness problems of controller for robot manipulator can not be objectively proved for the general desired trajectories because it has skipped crucial problems from trajectory dependent dynamics of manipulator by simple assumption. One of the expected problems in applications of the proposed control methodology is that simulation studies were tested only for a planar two-link manipulator.

Adaptive self-tuning controller for a robot manipulator was improved in

[Souissi&K. 87] and [Koivo&H. 91] by taking into consideration the interactions between joints of Stanford manipulator inside linear model for controller design. Both of them have used multi-input-multi-output autoregressive model with exogenous input, that is MIMO ARX-model, for the design of an one step ahead optimal controller. Main contribution of these schemes is that they include the effects of interactions between joints into system matrix in MIMO ARX model by estimating off-diagonal or blocked off-diagonal elements in square system matrix or input square matrix. This method, to a certain extent, improved convergence problem at moderate or high velocity after fine tuning the forgetting factor and weight constants for optimal controller along given desired velocity profile. But it still has the drawback of a large tracking error when desired trajectories are changed to even slower velocity profile. Thus, since it is very difficult or time-consuming to manually tune constants for controller for each desired trajectory, the method sometimes failed to converge to desired response of bounded-velocity profile. There is no test which involves picking up a payload.

1.3 Summary

The adaptive control of robot manipulators is particularly challenging because of coupling between joints, non-linearity, the trajectory dependent dynamics of manipulator and uncertainty of payload in conjunction with on-line implementation of the control laws. But as illustrated in the previous section, the control of a robot manipulator has common difficulties or problems as follows:

- The model used for most adaptive controller designs is linear although the real dynamics of manipulator is nonlinear.

- How to take coupling forces between joints into account in the design of controller.
- How to make the controller more robust about both slowly and fast varying desired trajectories without manual tuning constants in the adaptive controller law.
- How well does the adaptive controller cope with uncertainty of payload.
- How much is the simulation of a plant close to the real plant.

1.4 Objectives of Thesis

In this thesis, an improved MIMO adaptive self-tuning LQG controller for the six-joint PUMA manipulator is developed and tested using the complete six-joint PUMA dynamics. To the best knowledge of this author, no controllers have ever been tested on the simulated dynamics of all revolute six-joint manipulator. In order to show the adaptive capability of controller for robot manipulator, it is desirable for the controller to be tested on a model which would resemble the actual plant in respect of the crucial dynamics factor among motion of manipulator. The proposed controller is tested on moderate and very high velocity profile which generally cause difficulties of non-linearity, coupling forces between joints and convergence of controller. Very often, a controller which works well on fast varying desired trajectory, does not perform well on slowly varying reference signal. As an illustration, the adaptive self-tuning controller in [Koivo&H. 91] may not work properly on low velocity profile of desired trajectory although it works well on high velocity profile due to lack of adaptation capability for a slowly time-varying system. The test which includes picking up the unknown payload is also performed, and the effect of measurement error of white Gaussian zero-mean type is also simulated and then compensated properly.

1.5 Organization of Thesis

A general discussion about robotics is included in Chapter 2.

In Chapter 3, the Lagrange-Euler dynamics of n-series linked manipulator is shown with the derivation proceeding from basic newtonian mechanics to compact matrix form which used in the simulation of PUMA 600 in this thesis. The purpose of deriving the dynamics equation is to understand elementary terms of motion dynamics which is necessary for the design of controller.

The adaptive self-tuning LQG controller based on an autoregressive model with exogenous input is explained for the control of robot manipulator in Chapter 4.

Simulation results and improvements resulting from the use of the controller are presented in Chapter 5.

Finally, conclusions and suggestions for future research are in Chapter 6.

Chapter 2

Robot Systems - Background

In this Chapter general discussion of the robot systems is presented. In particular, the discussion will be developed in conjunction with PUMA 600 manipulator.

2.1 Description of Manipulator by D-H Notation

[Denavit&H. 55] first proposed a systematic method to define a local coordinate system for each link of a robot manipulator. The relations between a world coordinate system and a local coordinate system will be described by link parameters and joint variables from Denavit-Hartenberg(D-H) coordinate system(frame), which are often called *kinematic parameters*. A schematic picture of PUMA 600 manipulator is shown in Figure 1 where D-H coordinate system and its kinematic parameters are used.

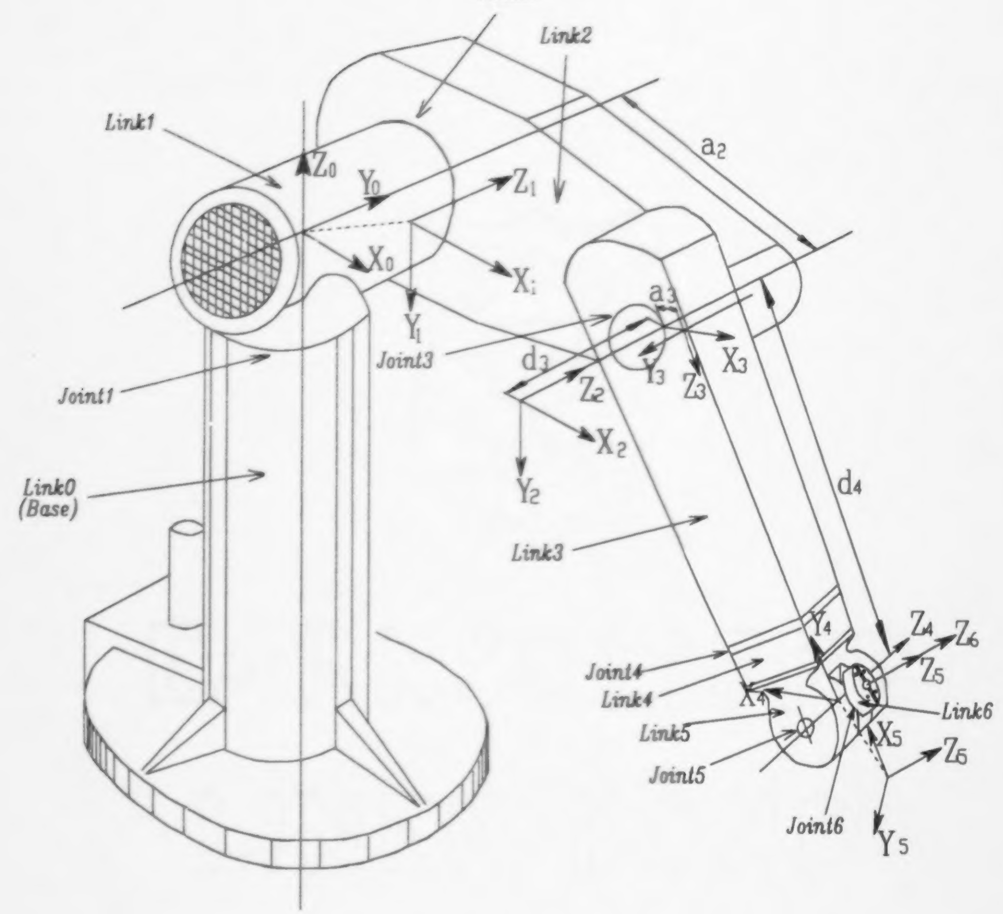
Joint2
Link2
Link1
a_2
Z_0
Y_0
Z_1
X_0
X_1
Y_1
a_3
Joint3
d_3
Y_3
Z_3
X_3
Z_2
Joint1
X_2
Y_2
d_4
Link0
(Base)
Link3
Y_4
Z_4
Z_6
Z_5
Joint4
Link4
X_4
Link6
Link5
X_5
Joint5
Z_6
Joint6
Y_5

Figure 1: PUMA Manipulator with D-H Notation

2.1.1 Determination of Coordinate System in Robotics

PUMA 600 series robot manipulator consists of seven links connected by revoluted six joints. Since the motion of the links involves angular rotation, the position and orientation of the end-effector has definite relationships with joint angles(or joint variables) and links. These mathematical relations depend on the coordinate systems chosen, and play an important role in the kinematics, dynamics and the control of robot manipulator. Since PUMA 600 has six joints and seven links, we can assign one coordinate system to each link. One coordinate system whose origin can be fixed to the base, or at the shoulder of the manipulator, is called the *World Coordinate System*, while the six coordinate systems whose origins can be assigned to the each of the six links are called *Local Coordinate Systems*. Accordingly the local coordinate system(frame) moves with the links.

The problem is to assign rectangular right-hand coordinate frames and kinematic parameters for the links. Then, the transformation matrices relating to the coordinate frames are to be written. For the next link $i+1$ with respect to link i in Figure 1, the coordinate axes are chosen by the D-H coordinate frame assigning method [Denavit&H. 55], [Bejczy 74], [Lee 82].

2.1.2 Determination of the Kinematic Parameters

After establishing the coordinate frames, it is necessary to define four kinematic parameters which are sometimes called the ***structural kinematic parameters*** since they depend on the structure of the given manipulator and describe the relative position of successive pair of the axes in two coordinate systems. Having established four structural parameters d_i, a_i, α_i, and θ_i, $i = 1, \cdots, n$ for a particular manipulator, the transformation matrix between the adjacent

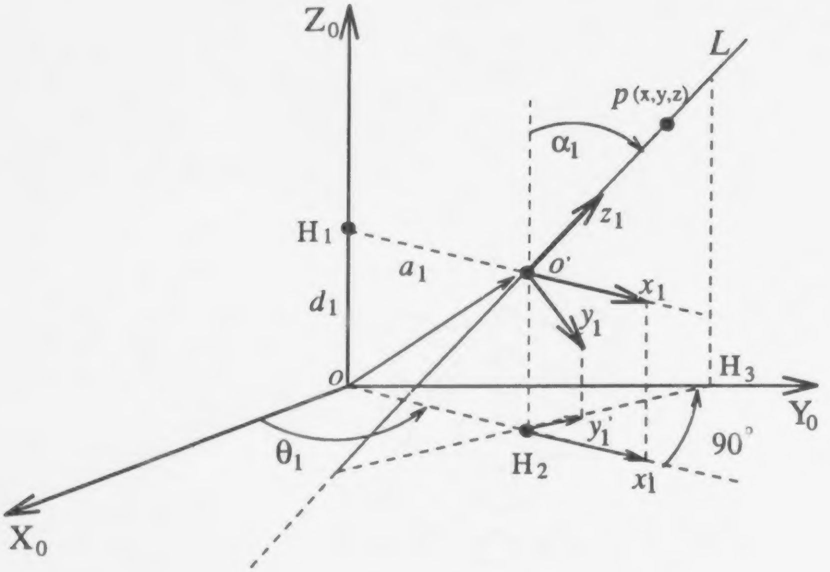

Figure 2: Illustration of Structural Parameters

coordinate frames may be written.

For the purpose of illustrating four structural parameters, a straight line such as L shown in Figure 2 can be described in terms of four parameters - *length* a_i, the *twist angle* α_i, *distance* d_i, and *angle* θ_i between links.

In Figure 2, since a straight line L is assumed to represent a rotational axis, a local coordinate frame is to be assigned. For a multiple joint serial link manipulator, the aforementioned structural parameters are determined for the ith link, $i = 1, \cdots, n$ as follows:

1. a_i : The distance from the origin of the ith coordinate frame to the intersection of the Z_{i-1}- and the X_i-axis along the x_i-axis.

2. α_i : The angle of rotation about the positive X_i-axis is measured from the positive Z_{i-1}(or its parallel projection) to the positive Z_i-axis, where the positive direction is counterclockwise.

joint $i =$	α_i°	θ_i°	d_i	a_i
1	-90	θ_1	0	0
2	0	θ_2	0	a_2
3	90	θ_3	d_3	a_3
4	-90	θ_4	d_4	0
5	90	θ_5	0	0
6	0	θ_6	0	0

Table 2.1: Structural Kinematic Parameters

3. θ_i : The angle of rotation about the positive Z_{i-1} is measured from the positive X_{i-1}-axis to the positive X_i-axis. It is positive in the counter-clockwise direction.

4. d_i : The distance from the origin of the $(i-1)$st coordinate frame to the intersection of the Z_{i-1}-axis, and the X_i-axis along the Z_{i-1}-axis.

Based on the procedures to determine the structural kinematic parameters, the values of d_i, a_i, α_i and θ_i [Paul&R.&Z. 83] can be obtained for PUMA 600 in Table 2.1 through Figure 1.

2.2 Determination of Transformation Matrix

A 4 × 4 matrix, which is called the ***transformation matrix***, relates the link-attached coordinate frame to the reference coordinate frame.

2.2.1 Recursive Transformation Matrix

If vector P_i is known in the ith coordinate frame, then it can be expressed in the $(i-1)$st coordinate frame as P_{i-1}, that is

$$P_{i-1} = A_{i-1}^{i} P_i \tag{2.1}$$

where matrix A_{i-1}^{i} of the *recursive transformation matrix* can be written by the following general form:

$$A_{i-1}^{i} = \left[\begin{array}{cccc|c} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & & d_i \\ \hline 0 & 0 & 0 & & 1 \end{array}\right] \tag{2.2}$$

We can extend the equation 2.1 as follows:

$$\begin{aligned} P_0 &= A_0^1 A_1^2 A_2^3 A_3^4 A_4^5 A_5^6 \, P_6 \\ &= A_0^6 \, P_6 \end{aligned} \tag{2.3}$$

Matrix A_0^6 in Equation 2.3 is a function of structural kinematic parameters. When these values are known, the position of the end-effector can be calculated by Equation 2.3. The determination of this position from the values of the joint variables is referred to as solving the *forward kinematic equations.*

2.2.2 Partition and Decomposition of Transformation Matrix

Transformation matrix A_0^6 can be *partitioned* into four parts as follows:

$$
\begin{aligned}
A_0^6 &= \left[\begin{array}{c|c} \text{Rotation Submatrix} & \text{Translation Vector} \\ (3 \times 3) & (3 \times 1) \\ \hline \text{Perspective Vector} & \text{Scaling Factor} \\ (1 \times 3) & (1 \times 1) \end{array}\right] \\
&= \left[\begin{array}{ccc|c} X_x & Y_x & Z_x & P_x \\ X_y & Y_y & Z_y & P_y \\ X_z & Y_z & Z_z & P_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right]
\end{aligned}
\tag{2.4}
$$

And also transformation matrix A_0^6 can be ***decomposed*** into matrix multiplication of translation matrix-$Trans(P_x, P_y, P_z)$ and rotation matrix-$RPY(\Psi_x, \Psi_y, \Psi_z)$ as follows:

$$
\begin{aligned}
A_0^6 &= Trans(P_x, P_y, P_z)RPY(\Psi_x, \Psi_y, \Psi_z) \\
&= \left[\begin{array}{ccc|c} 1 & 0 & 0 & P_x \\ 0 & 1 & 0 & P_y \\ 0 & 0 & 1 & P_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \left[\begin{array}{ccc|c} X_x & Y_x & Z_x & 0 \\ X_y & Y_y & Z_y & 0 \\ X_z & Y_z & Z_z & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]
\end{aligned}
\tag{2.5}
$$

2.3 Orientation of End-Effector

The grasping the object by end-effector means that the position and orientation of end-effector are the same as those of object in reference coordinate frame. The simplest one to understand is roll(Ψ_z), pitch(Ψ_y), and yaw(Ψ_x) in Figure 3 which are called by *Euler Angles*. These R-P-Y angles are used to represent the overall orientation of the object or end-effector of the manipulator.

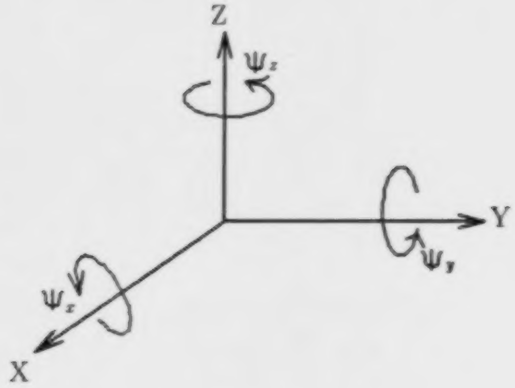

Figure 3: Euler Angles for Orientation

A sequence of three rotations is a rotation about X axis, a rotation about the Y axis, and a rotation about the Z axis. So rotation matrix, $RPY(\Psi_x, \Psi_y, \Psi_z)$ can be performed by means of successive rotational operations:

$$RPY(\Psi_z, \Psi_y, \Psi_x) = Rot(z, \Psi_z)Rot(y, \Psi_y)Rot(x, \Psi_x) =$$

$$\left[\begin{array}{ccc|c} c(\Psi_z)c(\Psi_y) & c(\Psi_z)s(\Psi_y)s(\Psi_x) - s(\Psi_z)c(\Psi_x) & c(\Psi_z)s(\Psi_y)c(\Psi_x) + s(\Psi_z)s(\Psi_x) & 0 \\ s(\Psi_z)c(\Psi_y) & s(\Psi_z)s(\Psi_y)s(\psi_x) + c(\Psi_z)c(\Psi_x) & s(\Psi_z)s(\Psi_y)c(\Psi_x) - c(\Psi_z)s(\Psi_x) & 0 \\ -s(\Psi_y) & c(\Psi_y)s(\Psi_x) & c(\Psi_y)c(\Psi_x) & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{2.6}$$

where

$$Rot(z, \Psi_z) = \begin{bmatrix} c(\Psi_z) & -s(\Psi_z) & 0 & 0 \\ s(\Psi_z) & c(\Psi_z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Rot(y, \Psi_y) = \begin{bmatrix} c(\Psi_y) & 0 & s(\Psi_y) & 0 \\ 0 & 1 & 0 & 0 \\ -s(\Psi_y) & 0 & c(\Psi_y) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Rot(x, \Psi_x) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c(\Psi_x) & -s(\Psi_x) & 0 \\ 0 & s(\Psi_x) & c(\Psi_x) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $s(\cdot)$ and $c(\cdot)$ stand for $\sin(\cdot)$ and $\cos(\cdot)$ respectively. Above $Rot(\cdot)$ matrices are called the ***basic homogeneous rotation matrices***.

2.3.1 An Algorithm for Calculating Ψ_x, Ψ_y, and Ψ_z

R-P-Y angles for the orientation of manipulator need to be computed with given four structural parameters. Transformation matrix A_0^6 for PUMA robot manipulator can be calculated by successive multiplication of transformation matrix in equation 2.2 with structural kinematic parameters in Table 2.1. This successive matrix multiplication can be easily performed by using symbolic computation software, for example, *Maple V* developed by the University of Waterloo [Maple 91]. This result is equated by equation 2.4 for calculating $Roll(\Psi_z)$,$Pitch(\Psi_y)$, and $Yaw(\Psi_x)$ angles, i.e.,

$$\left.\begin{aligned}
X_x &= c_6c_5c_4c_1c_2c_3 - c_6c_5c_4c_1s_2s_3 - c_6c_5s_1s_4 - c_6s_5c_1c_2s_3 - c_6s_5c_1s_2c_3 \\
&\quad -s_6s_4c_1c_2c_3 + s_6s_4c_1s_2s_3 - s_6s_1c_4 \\
Y_x &= -s_6c_5 - c_4c_1c_2c_3 + s_6c_5c_4c_1s_2s_3 + s_6c_5s_1s_4 + s_6s_5c_1c_2s_3 \\
&\quad +s_6s_5c_1s_2c_3 - c_6s_4c_1c_2c_3 + c_6s_4c_1s_2s_3 - c_6s_1c_4 \\
Z_x &= s_5c_4c_1c_2c_3 - s_5c_4c_1s_2s_3 - s_5s_1s_4 + c_5c_1c_2s_3 + c_5c_1s_2c_3 \\
P_x &= d_4c_1c_2s_3 + d_4c_1s_2c_3 + c_1c_2a_3c_3 - c_1s_2a_3s_3 - s_1d_3 + c_1a_2c_2 \\[1ex]
X_y &= c_6c_5c_4s_1c_2c_3 - c_6c_5c_4s_1s_2s_3 + c_6c_5c_1s_4 - c_6s_5s_1c_2s_3 - c_6s_5s_1s_2c_3 \\
&\quad s_6s_4s_1c_2c_3 + s_6s_4s_1s_2s_3 + s_6c_1c_4 \\
Y_y &= -s_6c_5c_4s_1c_2c_3 + s_6c_5c_4s_1s_2s_3 - s_6c_5c_1s_4 + s_6s_5s_1c_2s_3 \\
&\quad -c_6s_4s_1c_2c_3 + c_6s_4s_1s_2s_3 + c_6c_1c_4 + s_6s_5s_1s_2c_3 \\
Z_y &= s_5c_4s_1c_2c_3 - s_5c_4s_1s_2s_3 + s_5c_1s_4 + c_5s_1c_2s_3 + c_5s_1s_2c_3 \\
P_y &= d_4s_1c_2s_3 + d_4s_1s_2c_3 + s_1c_2a_3c_3 - s_1s_2a_3s_3 + c_1d_3 + s_1a_2c_2 \\[1ex]
X_z &= -c_6c_4c_5s_2c_3 - c_6c_4c_5c_2s_3 + c_6s_5s_2s_3 - c_6s_5c_2c_3 + s_4s_6s_2c_3 \\
&\quad +s_4s_6c_2c_3 \\
Y_z &= s_6c_4c_5s_2c_3 + s_6c_4c_5c_2s_3 - s_6s_5s_2s_3 + s_6s_5c_2c_3 + s_4c_6s_2c_3 \\
&\quad +s_4c_6c_2s_3 \\
Z_z &= -c_4s_5s_2c_3 - c_4s_5c_2s_3 - c_5s_2s_3 + c_5c_2c_3 \\
P_z &= -d_4s_2s_3 + d_4c_2c_3 - s_2a_3c_3 - c_2a_3s_3 - a_2s_2.
\end{aligned}\right\} \tag{2.7}$$

But transformation matrix A_0^6 can be rewritten in terms of *Translation Matrix* and basic *Rotation Matrices* about $X - Y - Z$ axes:

$$A_0^6 = Trans(P_x, P_y, P_z)Rot(z, \Psi_z)Rot(y, \Psi_y)Rot(x, \Psi_x) \tag{2.8}$$

By rearranging equation 2.8:

$$Rot^{-1}(z, \Psi_z)Trans^{-1}(P_x, P_y, P_z)A_0^6 = Rot(y, \Psi_y)Rot(x, \Psi_x) \tag{2.9}$$

Left hand side of equation 2.9 is:

$$Rot^{-1}(z, \Psi_z) Trans^{-1}(P_x, P_y, P_z) A_0^6$$

$$= \left[\begin{array}{ccc|c} c(\Psi_z) & s(\Psi_z) & 0 & 0 \\ -s(\Psi_z) & c(\Psi_z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \left[\begin{array}{ccc|c} 1 & 0 & 0 & -P_x \\ 0 & 1 & 0 & -P_y \\ 0 & 0 & 1 & -P_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \left[\begin{array}{ccc|c} X_x & Y_x & Z_x & P_x \\ X_y & Y_y & Z_y & P_y \\ X_z & Y_Z & Z_z & P_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

$$= \left[\begin{array}{ccc|c} X_x c(\Psi_z) + X_y s(\Psi_z) & Y_x c(\Psi_z) + Y_y s(\Psi_z) & Z_x c(\Psi_z) + Z_y s(\Psi_z) & 0 \\ X_y c(\Psi_z) - X_x s(\Psi_z) & Y_y c(\Psi_z) - Y_x s(\Psi_z) & Z_y c(\Psi_z) - Z_x s(\Psi_z) & 0 \\ X_z & Y_z & Z_z & P_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{2.10}$$

Right hand side of equation 2.9 is:

$$Rot(y, \Psi_y) Rot(x, \Psi_x)$$

$$= \left[\begin{array}{ccc|c} c(\Psi_y) & 0 & s(\Psi_y) & 0 \\ 0 & 1 & 0 & 0 \\ -s(\Psi_y) & 0 & c(\Psi_y) & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & c(\Psi_x) & -s(\Psi_x) & 0 \\ 0 & s(\Psi_x) & c(\Psi_x) & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{2.11}$$

$$= \left[\begin{array}{ccc|c} c(\Psi_y) & s(\Psi_y) s(\psi_x) & s(\Psi_y) c(\Psi_x) & 0 \\ 0 & c(\Psi_x) & -s(\Psi_x) & 0 \\ -s(\Psi_y) & c(\Psi_y) s(\Psi_x) & c(\Psi_y) c(\Psi_x) & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

By comparing equation 2.10 and equation 2.11, we can calculating Ψ_z, Ψ_y, and Ψ_x as follows:

$$\mapsto X_y c(\Psi_z) - X_x s(\Psi_z) = 0$$

$$\longrightarrow \underline{\Psi_z = atan2(X_y, X_x)} \tag{2.12}$$

$$\mapsto X_x \cos(\Psi_z) + X_y \sin(\Psi_z) = \cos(\Psi_y),\ X_z = -\sin(\psi_y)$$
$$\tan(\Psi_y) = \frac{\sin(\Psi_y)}{\cos(\Psi_y)} = \frac{-X_z}{X_x \cos(\Psi_z) + X_y \sin(\Psi_z)}$$

$$\longrightarrow \underline{\Psi_y = atan2(-X_z, X_x \cos(\Psi_z) + X_y \sin(\Psi_z)} \tag{2.13}$$

$$\mapsto Y_z = \cos(\Psi_y)\sin(\Psi_x),\ Z_z = \cos(\Psi_y)\cos(\Psi_x)$$
$$\tan(\Psi_x) = \frac{\sin(\psi_x)}{\cos(\psi_x)} = \frac{Y_z}{Z_z}$$

$$\longrightarrow \underline{\Psi_x = atan2(Y_z, Z_z)} \tag{2.14}$$

where argument values of *atan*2() in equations 2.12 - 2.14 are given in equation 2.7.

2.4 Summary - Need for Six-joint Dynamics

The position of the end-effector origin, i.e., P_x, P_y and P_z in equation 2.7 is only function of first three joint variables, θ_1, θ_2 and θ_3 instead of all six joint variables, $\theta_1, \cdots, \theta_6$ because other structural kinematic parameters are constants in Table 2.1. This fact is useful for the controller design such that the dynamics of the manipulator position depends on first three joint variables. Sometimes, the controller for the robot manipulator can use the three joint dynamics of the manipulator only for the position control. On the other hand, Euler angles, Ψ_x, Ψ_y, and Ψ_z in equations 2.12 - 2.14, which are representing the orientaton of the end-effector, depend on all six joint-variables. So, the controller for the end-effector orientation may need to use the dynamics of the six-joint manipulator.

Chapter 3

Modeling of Physical PUMA Manipulator : Dynamics Model

The dynamics equations or dynamics models describe the motion of a manipulator by means of differential or difference equations (or partial differential equations for robot manipulator with nonrigid links). The equations of motion are useful for computer simulation and the design of controller for a robot manipulator.

In this chapter, we shall present the dynamics model based on *Lagrange-Euler* equation and its customized closed-form for PUMA 600.

3.1 Lagrange-Euler(L-E) Dynamics Model

The derivation of the dynamics model of a robot manipulator based on the Lagrange-Euler equation is systematic [Uicker 65], [Bejczy 74], [Paul 81]. By assuming rigid body motion, the resulting model of motion is a set of second-order coupled nonlinear differential equations. Using the 4×4 transformation matrix representation of the kinematic chain and the Lagrangian equation, [Bejczy 74] has shown that the dynamic motion equations for a six degree-of-freedom Stanford robot manipulator are highly nonlinear and consist of inertia loading, coupling reaction forces between joints(i.e., Coriolis and centrifugal) and gravity loading effects. These torque/forces depend on the robot manipulator's physical parameters, instantaneous joint position, velocity and acceleration, and the load it is carrying.

The L-E dynamics model of motion provides explicit state equations for the manipulator dynamics and can be utilized to analyze and design advanced control strategies in joint-variable space. This L-E dynamics model is used to solve for the *forward dynamics* problem, that is, given the desired torque/forces, the equations of dynamics model are used to obtain the joint accelerations which are then integrated to solve for joint positions and their velocities; or for the *inverse dynamics* problem, that is, given the desired joint positions, velocities and accelerations, generalized forces/torque are computed.

Unfortunately, the computation of these coefficients, especially in case of six joint(d-o-f) robot manipulator, requires a fair amount of arithmetic operations even though its formulations are straightforward, systematic and readable. Thus, the L-E model is very difficult to utilize directly for real-time control purposes unless they are simplified. So, complete closed-form equations of models are seldom presented for robot manipulator more than three d-o-f manipulator although the literature abounds with formulations for generating complete dynamics robot models [Neuman&M. 85], [Brady 82].

However, [Neuman&M. 87] has performed complete and customized closed-form dynamics model of the six d-o-f PUMA robot manipulator in real-time using the symbolically processing software - ARM(Algebraic Robot Modeler), where the unmodeled dynamics like friction, backlash, dynamics of actuator, etc., exists.

[Neuman&M. 87]'s customized closed-form of dynamics model, which is based on L-E equation and customized with some parameters from [Paul&R.&Z. 83], and which has reduced the computational requirements, is useful for simulating six d-o-f PUMA robot manipulator on computer and precompensating the nonlinear terms in the design of adaptive controller. The customized closed-form equations of dynamics model for PUMA 600 robot manipulator, written in C language, will be presented in Appendix 7.

3.2 Derivation of L-E Dynamics Model for a Motion of Robot Manipulator

For control analysis and design, researchers would like to obtain an explicit set of closed-form of differential equations(or state equations) that describe the dynamic behavior of a robot manipulator. In addition, the interaction and coupling reaction forces in the equations should be easily identified so that a proper controller can be designed to compensate for their effects. Lagrange-Euler Dynamics model is one of the good models for the above purpose [Huston&K. 82].

3.2.1 Lagrange-Euler Equation

To determine a differential equation model for the motion of n d-o-f robot manipulator, the Lagrangian dynamics technique is employed. For a *conservative* system, *Lagrange-Euler equation* is shown as below:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathcal{L}}{\partial q_i} = 0 \qquad i = 1, \ldots, n \tag{3.1}$$

When external forces are acting on the system, they are included on the right hand side of equation 3.1. Thus, Lagrange-Euler equation for a *nonconservative* system is represented by the following form:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathcal{L}}{\partial q_i} = \tau_i \qquad i = 1, \ldots, n \tag{3.2}$$

where

$\mathcal{L}$ = Lagrange function = kinetic energy(K) − potential energy(P)

K = total kinetic energy of the robot manipulator

P = total potential energy of the robot manipulator

q_i = generalized coordinates of the robot manipulator

$\dot{q}_i$ = first time derivative of the generalized coordinate, q_i

τ_i = generalized force(or torque) applied to the system at joint i to the derive link i

In order to determine the Lagrange-Euler(L-E) dynamics model from Lagrange-Euler equation for a nonconservative system, one is required to properly choose a set of generalized coordinates. For example, $q_i \equiv \theta_i$ corresponds to generalized coordinates of joint variables which are defined in each of the 4×4 transformation matrices shown in equation 2.2.

The following derivation of L-E dynamics model of a n d-o-f robot manipulator is based on transformation matrices in Chapter 2.

3.2.2 Kinetic Energy of a Robot Manipulator

The Lagrange-Euler equation 3.2 requires knowledge of the kinetic energy(K) of the physical system, which in turn requires knowledge of the velocity of each joint. In this section, the velocity of a point fixed in link i will first be derived and the effects of the motion of other joints on all points in this link will be explored.

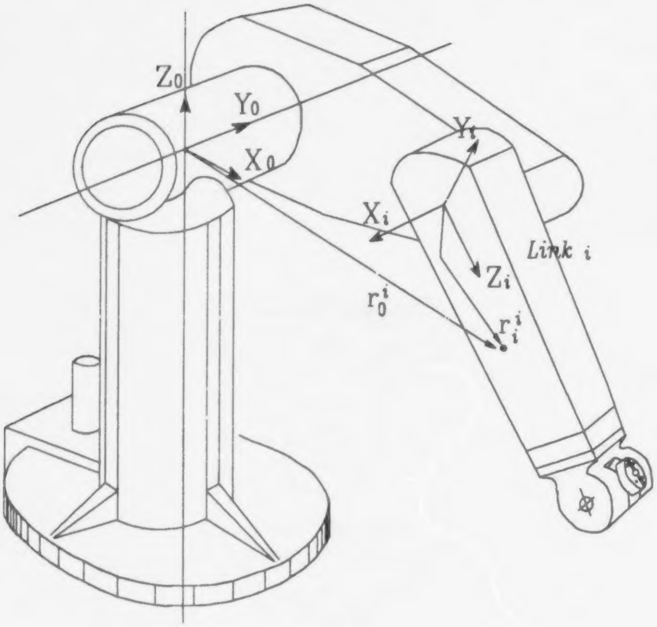

Figure 4: A Point $\mathbf{r}_i^i$ in Link i.

As shown in Figure 4, let $\mathbf{r}_i^i$ be a point in the link i, and $\mathbf{r}_0^i$ be the same point as $\mathbf{r}_i^i$ with respect to the base coordinate frame. The transformation matrix A_{i-1}^i relates the spatial displacement of the ith coordinate frame to the $(i-1)$th link coordinate frame as follows:

$$\mathbf{r}_i^i = \begin{bmatrix} x_i^i \\ y_i^i \\ z_i^i \\ 1 \end{bmatrix} = [x_i^i,\ y_i^i,\ z_i^i,\ 1]' \tag{3.3}$$

$$\mathbf{r}_0^i = \begin{bmatrix} x_0^i \\ y_0^i \\ z_0^i \\ 1 \end{bmatrix} = [x_0^i,\ y_0^i,\ z_0^i,\ 1]' \tag{3.4}$$

The velocity of $\mathbf{r}_0^i$ then is:

$$\dot{\mathbf{r}}_0^i = \frac{d}{dt}\left(\mathbf{r}_0^i\right) = \begin{bmatrix} \dot{x}_0^i \\ \dot{y}_0^i \\ \dot{z}_0^i \\ 1 \end{bmatrix} \tag{3.5}$$

$$\mathbf{r}_0^i = A_0^i r_i^i \tag{3.6}$$

where

$$A_0^i = A_0^1 A_1^2 \cdots A_{i-1}^i$$

The velocity of $\mathbf{r}_i^i$ expressed in the base coordinate frame can be expressed as:

$$\begin{aligned}
\dot{\mathbf{r}}_0^i &= \frac{d}{dt}(\mathbf{r}_0^i) \\
&= \frac{d}{dt}(A_0^i \mathbf{r}_i^i) \\
&= \frac{d}{dt}\left(A_0^1 A_1^2 \cdots A_{i-1}^i \mathbf{r}_i^i\right) \\
&= \frac{d}{dt}\left(\dot{A}_0^1 \cdots A_{i-1}^i \mathbf{r}_i^i + A_0^1 \dot{A}_1^2 \cdots \mathbf{r}_i^i + \cdots + A_0^1 \cdots \dot{A}_{i-1}^i \mathbf{r}_i^i + A_0^1 \cdots A_{i-1}^i \dot{\mathbf{r}}_i^i\right) \\
&= \left(\frac{\partial A_0^i}{\partial q_1}\dot{q}_1 + \frac{\partial A_0^i}{\partial q_2}\dot{q}_2 + \cdots + \frac{\partial A_0^i}{\partial q_i}\dot{q}_i\right)\mathbf{r}_i^i \\
&= \left(\sum_{j=1}^{i} \frac{\partial A_0^i}{\partial q_j}\dot{q}_j\right)\mathbf{r}_i^i \quad , \quad \text{where} \quad \dot{\mathbf{r}}_i^i = 0
\end{aligned} \tag{3.7}$$

After obtaining a point velocity of each link i, we need to find the kinetic energy of link i. Let K_0^i be the kinetic energy of link i expressed in the base coordinate frame, and dK_0^i be the kinetic energy of a particle with differential mass dm_i in link i, then the kinetic energy of the differential mass dm_i is:

$$\begin{aligned}
dK_0^i &= \frac{1}{2}\left((\dot{x}_0^i)^2 + (\dot{y}_0^i)^2 + (\dot{z}_0^i)^2\right) dm_i \\
&= \frac{1}{2}\left(\dot{\mathbf{r}}_0^i \cdot \dot{\mathbf{r}}_0^i\right) dm_i \\
&= \frac{1}{2} Tr\left[\dot{\mathbf{r}}_0^i (\dot{\mathbf{r}}_0^i)'\right] dm_i \\
&= \frac{1}{2} Tr\left[\sum_{j=1}^{i} \frac{\partial A_0^i}{\partial q_j} \dot{q}_j \mathbf{r}_i^i \left(\sum_{k=1}^{i} \frac{\partial A_0^i}{\partial q_k} \dot{q}_k \mathbf{r}_i^i\right)'\right] dm_i \\
&= \frac{1}{2} Tr\left[\sum_{j=1}^{i}\sum_{k=1}^{i} \frac{\partial A_0^i}{\partial q_j} \mathbf{r}_i^i \mathbf{r}_i^{i\prime} \frac{\partial A_0^{i\prime}}{\partial q_k} \dot{q}_j \dot{q}_k\right] dm_i \qquad (3.8)
\end{aligned}$$

where $Tr[\cdot]$, a trace operator instead of a vector dot product, is used in the above equation. Then, kinetic energy of link i is:

$$\begin{aligned}
K_0^i &= \int dK_0^i \\
&= \frac{1}{2} Tr\left[\sum_{j=1}^{i}\sum_{k=1}^{i} \frac{\partial A_0^i}{\partial q_j} \left(\int \mathbf{r}_i^i \mathbf{r}_i^{i\prime} dm_i\right) \frac{\partial A_0^{i\prime}}{\partial q_k} \dot{q}_j \dot{q}_k\right] \\
&= \frac{1}{2} Tr\left[\sum_{j=1}^{i}\sum_{k=1}^{i} \frac{\partial A_0^i}{\partial q_j} (J_i) \frac{\partial A_0^{i\prime}}{\partial q_k} \dot{q}_j \dot{q}_k\right] \qquad (3.9)
\end{aligned}$$

In equation 3.9, the integral can be put inside bracket since $\frac{\partial A_0^i}{\partial q_j}$(that is, the rate of change of the point($\mathbf{r}_i^i$) in link i relative to base coordinate frame as q_j changes) is constant for all points on link i and independent of the mass distribution of the link, and also $\dot{q}_i$ are independent of the mass distribution of link i. The integral term inside parenthesis in equation 3.9 known as *pseudo-inertia matrix* , J_i is represented by recalling $\mathbf{r}_i^i = [x_i^i \;\; y_i^i \;\; z_i^i \;\; 1]'$ as follows:

$$\begin{aligned}
J_i &= \int \mathbf{r}_i^i \mathbf{r}_i^{i\prime} dm_i \\
&= \begin{bmatrix} \int x_i^{i2} dm_i & \int x_i^i y_i^i dm_i & \int x_i^i z_i^i dm_i & \int x_i^i dm_i \\ \int x_i^i y_i^i dm_i & \int y_i^{i2} dm_i & \int y_i^i z_i^i dm_i & \int y_i^i dm_i \\ \int x_i^i z_i^i dm_i & \int y_i^i z_i^i dm_i & \int z_i^{i2} dm_i & \int z_i^i dm_i \\ \int x_i^i dm_i & \int y_i^i dm_i & \int z_i^i dm_i & \int dm_i \end{bmatrix} \\
&= \begin{bmatrix} \frac{-S_{ixx}^2 + S_{iyy}^2 + S_{izz}^2}{2} & S_{ixy}^2 & S_{ixz}^2 & \bar{x}_i^i \\ S_{ixy}^2 & \frac{S_{ixx}^2 - S_{iyy}^2 + S_{izz}^2}{2} & S_{iyz}^2 & \bar{y}_i^i \\ S_{ixz}^2 & S_{iyz}^2 & \frac{S_{ixx}^2 + S_{iyy}^2 - S_{izz}^2}{2} & \bar{z}_i^i \\ \bar{x}_i^i & \bar{y}_i^i & \bar{z}_i^i & 1 \end{bmatrix} m_i \\
&= \begin{bmatrix} \frac{-I_{ixx} + I_{iyy} + I_{izz}}{2} & I_{ixy} & I_{ixz} & m_i \bar{x}_i^i \\ I_{ixy} & \frac{I_{ixx} - I_{iyy} + I_{izz}}{2} & I_{iyz} & m_i \bar{y}_i^i \\ I_{ixz} & I_{iyz} & \frac{I_{ixx} + I_{iyy} - I_{izz}}{2} & m_i \bar{z}_i^i \\ m_i \bar{x}_i^i & m_i \bar{y}_i^i & m_i \bar{z}_i^i & m_i \end{bmatrix} \qquad (3.10)
\end{aligned}$$

where $\bar{\mathbf{r}}_i^i = [\bar{x}_i^i \;\; \bar{y}_i^i \;\; \bar{z}_i^i \;\; 1]'$ is the center of mass vector of link i from the ith link coordinate frame; S_{ixy} is called the *radius of gyration* of link i about $X_i - Y_i$ axes; *inertia tensor* I_i and *first moments of body* are defined as:

- **Inertia Tensor I_i**

  1. Moments of Inertia

$$I_{ixx} = \int \left(y_i^{i2} + z_i^{i2} \right) dm_i$$

$$I_{iyy} = \int \left(x_i^{i2} + z_i^{i2} \right) dm_i$$

$$I_{izz} = \int \left(x_i^{i2} + y_i^{i2} \right) dm_i$$

2. Cross-Products of Inertia (Symmetry : $I_{ixy} = I_{iyx}$)

$$I_{ixy} = \int x_i^i y_i^i dm_i$$

$$I_{ixz} = \int x_i^i z_i^i dm_i$$

$$I_{iyz} = \int y_i^i z_i^i dm_i$$

- **First Moments of Body**

$$m\bar{x}_i^i = \int x_i^i dm_i$$

$$m\bar{y}_i^i = \int y_i^i dm_i$$

$$m\bar{z}_i^i = \int z_i^i dm_i$$

Hence, the total kinematic energy of the manipulator from equation 3.9 is:

$$\begin{aligned} K &= \sum_{i=1}^{n} K_0^i \\ &= \sum_{i=1}^{n} \left\{ \frac{1}{2} Tr \left[\sum_{j=1}^{i} \sum_{k=1}^{i} \frac{\partial A_0^i}{\partial q_j} J_i \frac{\partial A_0^{i\,\prime}}{\partial q_k} \dot{q}_j \dot{q}_k \right] \right\} \\ &= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{i} \sum_{k=1}^{i} Tr \left[\frac{\partial A_0^i}{\partial q_j} J_i \frac{\partial A_0^i}{\partial q_k} \right] \dot{q}_j \dot{q}_k \end{aligned} \tag{3.11}$$

The total kinetic energy K is *scalar*, and the off-diagonal elements of J_i are zero because the classical cross-products of inertia are assumed to be zero[Paul&R.&Z. 83].

3.2.3 Potential Energy of a Robot Manipulator

Let the total potential energy of a robot manipulator be P and each of ith link's potential energy be P_i:

$$\begin{aligned} P_i &= -m_i \mathbf{g}\,(\bar{\mathbf{r}}_0^i) \\ &= -m_i \mathbf{g}\,(A_C^i \bar{\mathbf{r}}_i^i) \qquad i = 1, 2, \ldots, n \qquad (3.12) \end{aligned}$$

where $\bar{\mathbf{r}}_0^i = [\bar{x}_0^i \ \ \bar{y}_0^i \ \ \bar{z}_0^i]'$ and $\bar{\mathbf{r}}_i^i = [\bar{x}_i^i \ \ \bar{y}_i^i \ \ \bar{z}_i^i]'$.

The total potential energy of the robot manipulator can be obtained by summing all the potential energies in each link:

$$\begin{aligned} P &= \sum_{i=1}^{n} P_i \\ &= \sum_{i=1}^{n} -m_i \mathbf{g}(A_0^i \bar{\mathbf{r}}_i^i) \qquad (3.13) \end{aligned}$$

where $\mathbf{g} = [g_x,\ g_y,\ g_z,\ 0]$ is a gravity row vector expressed in the base coordinate system. For a level system, $\mathbf{g} = [0,\ 0,\ -|g|,\ 0]$ and g is the gravitational constant($g = 9.8062 m/\sec^2$).

3.2.4 L-E Dynamics Equations of a Robot Manipulator

Lagrangian function $\mathcal{L} = K - P$ can now formed from the kinetic energy and the potential energy, i.e.,

$$\begin{aligned} \mathcal{L} &= \{K\} - \{P\} \\ &= \left\{ \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{i} \sum_{k=1}^{i} Tr \left[\frac{\partial A_0^i}{\partial q_j} J_i \frac{\partial A_0^i}{\partial q_k} \right] \dot{q}_j \dot{q}_k \right\} - \left\{ \sum_{i=1}^{n} -m_i \mathbf{g}(A_0^i \bar{\mathbf{r}}_i^i) \right\} \quad (3.14) \end{aligned}$$

We now obtain the dynamics equations of a robot manipulator from below the Lagrange-Euler equation.

$$\tau_i = \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathcal{L}}{\partial q_i} \qquad i = 1, \ldots, n \tag{3.15}$$

Evaluaton of $\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_p}\right)$

Let's perform first differentiation of $\frac{\partial \mathcal{L}}{\partial \dot{q}_p}$:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \dot{q}_p} &= \frac{\partial}{\partial \dot{q}_p}\left\{\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=1}^{i} Tr\left[\frac{\partial A_0^i}{\partial q_j} J_i \frac{\partial A_0^i}{\partial q_k}'\right]\dot{q}_j\dot{q}_k + \sum_{i=1}^{n} m_i \mathbf{g}(A_0^i \bar{\mathbf{r}}_i^i)\right\} \\
&= \frac{1}{2}\sum_{i=1}^{n}\frac{\partial}{\partial \dot{q}_p}\left\{\sum_{j=1}^{i}\sum_{k=1}^{i} Tr\left[\frac{\partial A_0^i}{\partial q_j} J_i \frac{\partial A_0^i}{\partial q_k}'\right]\dot{q}_j\dot{q}_k\right\} \\
&\quad + \frac{\partial}{\partial \dot{q}_p}\left\{m_1 \mathbf{g} A_0^1 \bar{\mathbf{r}}_1^1 + m_2 \mathbf{g} A_0^2 \bar{\mathbf{r}}_2^2 + \cdots + m_n \mathbf{g} A_0^n \bar{\mathbf{r}}_n^n\right\} \\
&= \frac{1}{2}\sum_{i=1}^{n}\left\{\sum_{j=1}^{i} Tr\left[\frac{\partial A_0^i}{\partial q_j} J_i \frac{\partial A_0^i}{\partial q_p}'\right]\dot{q}_j + \frac{1}{2}\sum_{k=1}^{i} Tr\left[\frac{\partial A_0^i}{\partial q_p} J_i \frac{\partial A_0^i}{\partial q_k}'\right]\dot{q}_k\right\} \\
&\quad + 0 \qquad \left(\text{since } \frac{\partial A_0^i}{\partial \dot{q}_p} = 0\right) \\
&= \frac{1}{2}\sum_{i=1}^{n}\underbrace{\sum_{j=1}^{i} Tr\left[\frac{\partial A_0^i}{\partial q_j} J_i \frac{\partial A_0^i}{\partial q_p}'\right]\dot{q}_j}_{(a)} + \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{i} Tr\left[\frac{\partial A_0^i}{\partial q_p} J_i \frac{\partial A_0^i}{\partial q_k}'\right]\dot{q}_k
\end{aligned}$$

By changing the dummy index j to k at (a),

$$= \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{i} Tr\left[\frac{\partial A_0^i}{\partial q_k} J_i \frac{\partial A_0^i}{\partial q_p}'\right]\dot{q}_k + \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{i}\underbrace{Tr\left[\frac{\partial A_0^i}{\partial q_p} J_i \frac{\partial A_0^i}{\partial q_k}'\right]}_{(b)}\dot{q}_k$$

Since $Tr\left[\frac{\partial A_0^i}{\partial q_p} J_i \frac{\partial A_0^i}{\partial q_k}'\right] = Tr\left[\frac{\partial A_0^i}{\partial q_p} J_i \frac{\partial A_0^i}{\partial q_k}'\right]' = Tr\left[\frac{\partial A_0^i}{\partial q_k} J_i \frac{\partial A_0^i}{\partial q_p}'\right]$ at (b),

$$= \begin{cases} \sum_{i=1}^{n}\sum_{k=1}^{i} Tr\left[\frac{\partial A_0^i}{\partial q_k} J_i \frac{\partial A_0^i{}'}{\partial q_p}\right]\dot{q}_k & \text{for } p \leq i \\ 0 & \text{for } p > i \text{ since } \frac{\partial A_0^i}{\partial p} = 0 \end{cases} \tag{3.16}$$

We obtain:

$$\frac{\partial \mathcal{L}}{\partial \dot{q}_p} = \sum_{i=p}^{n}\sum_{k=1}^{i} Tr\left[\frac{\partial A_0^i}{\partial q_k} J_i \frac{\partial A_0^i{}'}{\partial q_p}\right]\dot{q}_k \tag{3.17}$$

Hence,

$$\begin{aligned} \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_p}\right) &= \frac{d}{dt}\left(\sum_{i=p}^{n}\sum_{k=1}^{i} Tr\left[\frac{\partial A_0^i}{\partial q_k} J_i \frac{\partial A_0^i{}'}{\partial q_p}\right]\dot{q}_k\right) \\ &= \sum_{i=p}^{n}\sum_{k=1}^{i} Tr\left[\frac{\partial A_0^i}{\partial q_k} J_i \frac{\partial A_0^i{}'}{\partial q_p}\right]\ddot{q}_k \\ &\quad + \sum_{i=p}^{n}\sum_{k=1}^{i}\sum_{m=1}^{i} Tr\left[\frac{\partial^2 A_0^i}{\partial q_k \partial q_m} J_i \frac{\partial A_0^i{}'}{\partial q_p}\right]\dot{q}_k\dot{q}_m \\ &\quad + \sum_{i=p}^{n}\sum_{k=1}^{i}\sum_{m=1}^{i} Tr\left[\frac{\partial^2 A_0^i}{\partial q_p \partial q_m} J_i \frac{\partial A_0^i{}'}{\partial q_k}\right]\dot{q}_k\dot{q}_m \end{aligned} \tag{3.18}$$

Evaluation of $\frac{\partial \mathcal{L}}{\partial q_p}$

The last term of Lagrange-Euler equation is evaluated, that is :

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial q_p} &= \frac{\partial}{\partial q_p}\left(\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=1}^{i} Tr\left[\frac{\partial A_0^i}{\partial q_j} J_i \frac{\partial A_0^i{}'}{\partial q_k}\right]\dot{q}_j\dot{q}_k + \sum_{i=1}^{n} m_i \mathbf{g} A_0^i \bar{\mathbf{r}}_i^i\right) \\ &= \frac{1}{2}\sum_{i=p}^{n}\sum_{j=1}^{i}\sum_{k=1}^{i} Tr\left[\frac{\partial^2 A_0^i}{\partial q_j \partial q_p} J_i \frac{\partial A_0^i{}'}{\partial q_k}\right]\dot{q}_j\dot{q}_k \\ &\quad + \sum_{i=p}^{n}\sum_{j=1}^{i}\sum_{k=1}^{i} \underbrace{Tr\left[\frac{\partial^2 A_0^i}{\partial q_k \partial q_p} J_i \frac{\partial A_0^i{}'}{\partial q_j}\right]}_{(a)}\dot{q}_j\dot{q}_k \end{aligned}$$

$$+\sum_{i=p}^{n} m_i \mathbf{g} \frac{\partial A_0^i}{\partial q_p} \bar{\mathbf{r}}_i^i$$

By interchanging the dummy indices of j and k at (a),

$$= \sum_{i=p}^{n} \sum_{j=1}^{i} \sum_{k=1}^{i} Tr \left[\frac{\partial^2 A_0^i}{\partial q_p \partial q_j} J_i \frac{\partial A_0^{i\prime}}{\partial q_k} \right] \dot{q}_j \dot{q}_k + \sum_{i=p}^{n} m_i \mathbf{g} \frac{\partial A_0^i}{\partial q_p} \bar{\mathbf{r}}_i^i$$

By changing j to m and then swapping $\sum_{m=1}^{i}$ and $\sum_{k=1}^{i}$,

$$= \sum_{i=p}^{n} \sum_{k=1}^{i} \sum_{m=1}^{i} Tr \left[\frac{\partial^2 A_0^i}{\partial q_p \partial q_m} J_i \frac{\partial A_0^{i\prime}}{\partial q_k} \right] \dot{q}_k \dot{q}_m + \sum_{i=p}^{n} m_i \mathbf{g} \frac{\partial A_0^i}{\partial q_p} \bar{\mathbf{r}}_i^i \qquad (3.19)$$

Evaluation of $\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_p}\right) - \frac{\partial \mathcal{L}}{\partial q_p}$

From equations 3.18 and 3.19, we obtain: the third term of $\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_p}\right)$ in equation 3.18 cancels the first term of $\frac{\partial \mathcal{L}}{\partial q_p}$ in equation 3.19.

$$\begin{aligned} \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_p}\right) - \frac{\partial \mathcal{L}}{\partial q_p} &= \sum_{i=p}^{n} \sum_{k=1}^{i} Tr \left[\frac{\partial A_0^i}{\partial q_k} J_i \frac{\partial A_0^{i\prime}}{\partial q_p} \right] \ddot{q}_k \\ &+ \sum_{i=p}^{n} \sum_{k=1}^{i} \sum_{m=1}^{i} Tr \left[\frac{\partial^2 A_0^i}{\partial q_k \partial q_m} J_i \frac{\partial A_0^{i\prime}}{\partial q_p} \right] \dot{q}_k \dot{q}_m \\ &- \sum_{i=1}^{n} m_i \mathbf{g} \frac{\partial A_0^i}{\partial q_p} \bar{\mathbf{r}}_i^i \qquad (3.20) \end{aligned}$$

Finally, we obtain the *dynamics equation* of a robot manipulator by changing dummy summation indices p to i and i to j:

$$\begin{aligned} \tau_i &= \sum_{j=i}^{n} \sum_{k=1}^{j} Tr \left[\frac{\partial A_0^j}{\partial q_k} J_j \frac{\partial A_0^{j\prime}}{\partial q_i} \right] \ddot{q}_k \\ &+ \sum_{j=i}^{n} \sum_{k=1}^{j} \sum_{m=1}^{j} Tr \left[\frac{\partial^2 A_0^j}{\partial q_k \partial q_m} J_j \frac{\partial A_0^{j\prime}}{\partial q_i} \right] \dot{q}_k \dot{q}_m \end{aligned}$$

$$-\sum_{j=i}^{n} m_j \mathbf{g} \frac{\partial A_0^j}{\partial q_i} \bar{\mathbf{r}}_j^j \tag{3.21}$$

3.2.5 Computational Simplification of Dynamics Equation using $Q - matrix$

The computation of the ***matrix partial derivatives*** in equation 3.21 is very time consuming. The calculations can be made faster by first noticing that transformation matrix A_{i-1}^i is a function of the generalized coordinate q_i only. So, the computation of partial derivative $\partial A_{i-1}^i / \partial q_i$ for serial link manipulators can be converted to a multiplication of matrices [Bejczy 74].

If we define $Q_i - matrix$ for a revolute joint i as below:

$$Q_i \equiv \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.22}$$

Then,

$$\frac{\partial A_{i-1}^i}{\partial q_i} = Q_i A_{i-1}^i \tag{3.23}$$

For example:

$$\frac{\partial A_{i-1}^i}{\partial \theta_i} = \begin{bmatrix} -\sin\theta_i & -\cos\alpha_i \cos\theta_i & \sin\alpha_i \cos\theta_i & -a_i \sin\theta_i \\ \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \cos\theta_i & a_i \cos\theta_i \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$
\begin{aligned}
&= \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}
\begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= Q_i A^i_{i-1}
\end{aligned}
$$

Hence, for $i = 1, 2, \ldots, n$:

$$
\begin{aligned}
\frac{\partial A_0^i}{\partial q_j} &= \frac{\partial}{\partial q_j}\left(\underbrace{A_0^1 A_1^2 \cdots A_{j-2}^{j-1}}\ \underbrace{A_{j-1}^j}\ \underbrace{A_j^{j+1} \cdots A_{i-1}^i}\right) \\
&= \overbrace{A_0^1 A_1^2 \ldots A_{j-2}^{j-1}}\ \overbrace{Q_j A_{j-1}^j}\ \overbrace{A_j^{j+1} \cdots A_{i-1}^i} \\
&= \begin{cases} A_0^{j-1} Q_j A_{j-1}^i & \text{for } 1 \le j \le i \\ 0 & \text{for } j > i \end{cases}
\end{aligned}
\tag{3.24}
$$

In order to simplify partial derivative notations of dynamics equation 3.21, we define the $U - matrix$ as follows:

$$
U_{ij} \equiv \frac{\partial A_0^i}{\partial q_j} = \begin{cases} A_0^{j-1} Q_j A_{j-1}^i & \text{for } j \le i \\ 0 & \text{for } j > i \end{cases} \tag{3.25}
$$

$$
\begin{aligned}
U_{ijk} &\equiv \frac{\partial U_{ij}}{\partial q_k} \\
&= \frac{\partial^2 A_0^i}{\partial q_k \partial q_j} = \begin{cases} A_0^{j-1} Q_j A_{j-1}^{k-1} Q_k A_{k-1}^i & \text{for } i \ge k \ge j \\ A_0^{k-1} Q_k A_{k-1}^{j-1} Q_j A_{j-1}^i & \text{for } i \ge j \ge k \\ 0 & \text{for } i < j \text{ or } i < k \end{cases}
\end{aligned}
\tag{3.26}
$$

For example, for a manipulator with all revolute joints $i = j = 1$ and $q_1 = \theta_1$,

$$
U_{111} = \frac{\partial U_{11}}{\partial \theta_1} = \frac{\partial}{\partial \theta_1}\left(Q_1 A_0^1\right) = Q_1 Q_1 A_0^1
$$

equation 3.26 can be interpreted as the interaction effects of the motion of joint j and joint k on all the points in link i. Hence, dynamics equation 3.21 is rewritten without partial derivative operations by using $U - matrix$:

$$\begin{aligned}\tau_i \; = \; & \sum_{j=i}^{n}\sum_{k=1}^{j} Tr[U_{jk}J_jU'_{ji}]\ddot{q}_k + \sum_{j=i}^{n}\sum_{k=1}^{j}\sum_{m=1}^{j} Tr[U_{jkm}J_jU'_{ji}]\dot{q}_k\dot{q}_m - \sum_{j=i}^{n} m_j \mathbf{g} U_{ji}\bar{\mathbf{r}}_j^j \\ & \text{for } i = 1, \ldots, n \qquad (3.27)\end{aligned}$$

The above equation can be expressed in a much simpler form as:

$$\tau_i = \sum_{k=1}^{n} D_{ik}\ddot{q}_k + \sum_{k=1}^{n}\sum_{m=1}^{n} c_{ikm}\dot{q}_k\dot{q}_m + G_i \qquad i = 1, \ldots, n \qquad (3.28)$$

Or in a matrix form:

$$\boldsymbol{\tau}(t) = \mathbf{D}(\mathbf{q}(t))\ddot{\mathbf{q}}(t) + \mathbf{C}(\mathbf{q}(t), \dot{\mathbf{q}}(t)) + \mathbf{G}(\mathbf{q}(t)) \qquad (3.29)$$

where

$\boldsymbol{\tau}(t) = n \times 1$ generalized torque vector applied at joint $i = 1, \ldots, n$, and can be expressed as:

$$\boldsymbol{\tau}(t) = [\tau_1(t),\ \tau_2(t), \ldots, \tau_n(t)]'$$

$\mathbf{q}(t) = n \times 1$ vector of the joint variables of the robot manipulator and can be expressed as:

$$\mathbf{q}(t) = [q_1(t),\ q_2(t), \ldots, q_n(t)]'$$

$\dot{\mathbf{q}}(t) = n \times 1$ vector of the joint velocity of the robot manipulator and can be expressed as:

$$\dot{\mathbf{q}} = [\dot{q}_1(t),\ \dot{q}_2(t), \ldots, \dot{q}_n(t)]'$$

$\ddot{\mathbf{q}}(t) = n \times 1$ vector of the acceleration of the joint variable $\mathbf{q}(t)$ and can be expressed as:

$$\ddot{\mathbf{q}} = [\ddot{q}_1(t),\ \ddot{q}_2(t), \ldots, \ddot{q}_n(t)]'$$

$\mathbf{D}(\mathbf{q}) = n \times n$ inertial acceleration-related symmetric matrix whose elements are:

$$D_{ik} = \sum_{j=\max(i,k)}^{n} Tr[U_{jk} J_j U'_{ji}] \qquad i,k = 1,\ldots,n$$

$\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) = n \times 1$ nonlinear Coliolis and centrifugal force vector whose elements are:

$$\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) = [C_1, C_2, \ldots, C_n]'$$

where

$$\begin{aligned} C_i &= \sum_{k=1}^{n}\sum_{m=1}^{n} c_{ikm}\dot{q}_k\dot{q}_m \qquad i = 1,\ldots,n \\ c_{ikm} &= \sum_{j=\max(i,k,m)}^{n} Tr[U_{jkm} J_j U'_{ji}] \quad i,k,m = 1,\ldots,n \end{aligned}$$

$\mathbf{G}(\mathbf{q}) = n \times 1$ gravity loading force vector whose elements are:

$$\mathbf{G}(\mathbf{q}) = [G_1, G2, \ldots, G_n]'$$

where

$$G_i = \sum_{j=i}^{n}(-m_j \mathbf{g} U_{ji} \bar{\mathbf{r}}_j^j) \qquad i = 1,\ldots,n$$

3.3 Customized Closed-form of Dynamics Equation for Six d-o-f PUMA Robot

For controlling the robot manipulator or simulating its behavior on computer, the dynamics coefficients of in equation 3.29, that is, $\mathbf{D}(\mathbf{q}(t)), \mathbf{C}(\mathbf{q}(t), \dot{\mathbf{q}}(t))$ and $\mathbf{G}(\mathbf{q}(t))$, need to be computed in real time. However, the algebraic manipulations leading to the complete dynamic model for robot manipulator become tedious and time-consuming as the number n of d-o-f increases, although formulation of equation 3.29 is inherently straightforward [Neuman&M. 85], [Bejczy 83], [Walker&O. 82]. So, complete dynamics equations are seldom presented for robot manipulators with more than three d-o-f although formulations for generating complete dynamic robot models have been shown in the literatures [Neuman&M. 85], [Brady 82]. The following section is to show how to build the customized closed-form dynamics equation based on Lagrange-Euler dynamics model of equation 3.29.

3.3.1 Customizing Dynamics Equation

In contrast to generalized-purpose algorithms, the practical problem with customized algorithms is that a different algorithm is required for a specified robot manipulator to reduce the computational requirements of manipulator dynamics for real-time control.

Computational savings of customized algorithms stem from kinematic and dynamic structure of the manipulator and systematic organization of the symbolic model when the symbolically processing computer program is used [Murray&N. 84], [Neuman&M. 87].

For example, Algebraic Robot Modeler(ARM) [Neuman&M. 87] resolves this problem as follows: The kinematic and dynamic structure of the manipulator is exploited during the *off-line* symbolic modeling as additions of zero and multiplications by ±1/zero are canceled algebraically. Upon completing the modeling stage, ARM applies a few elementary grouping and factoring rules to the algebraic expressions in closed-form dynamic robot models to remove repetitive calculations within and across equations.

On the other hand, the dynamic coefficients D_{ik} and c_{ikm} in equation 3.28 exhibit the symmetric reflective coupling properties [Tourassis&N. 85], [Lewis 74].

- Symmetry: $D_{ik} = D_{ki}$ and $c_{ikm} = c_{imk}$.
- Reflective coupling: $c_{ikm} = -c_{mki}$ for $k \leq i$ and m.

The dynamics coefficients D_{ik}, $c_i km$, and G_i depend on the constant geometrical, inertial, and gravitational parameters of the robot manipulator. The symbolically processing software, which produces the closed-form of dynamics robot equations, accepts these constant parameters as input, either symbolically or numerically at the user's option.

The following section describes the parameters for the [Paul&R.&Z. 83] with which ARM generates the complete customized closed-form of dynamics equation of PUMA 600 robot manipulator. This dynamics equation is useful for simulating the motion of physical robot and thus, is used in simulating the plant of PUMA robot manipulator on computer in this thesis.

3.3.2 Parameters of PUMA 600

The six d-o-f revolute PUMA robot has parallel/perpendicular joint axes, diagonal link inertia tensor, sparse center-of-mass vector, and six link coordinate frames assigned in Chapter 2. Physical parameter values for PUMA 600 are shown as follows:

Kinematic Link Parameters

The transformation matrix A_{i-1}^{i} for the dynamics equation is evaluated with table 3.2 which has been determined in Chapter 2.

$$A_{i-1}^{i} = \left[\begin{array}{ccc|c} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

joint $i =$	α_i°	θ_i°	d_i	a_i
1	−90	θ_1	0	0
2	0	θ_2	0	a_2=43.2cm
3	90	θ_3	d_3=12.5cm	a_3=1.9cm
4	−90	θ_4	d_4=43.2cm	0
5	90	θ_5	0	0
6	0	θ_6	0	0

Table 3.2: Kinematic Link Parameters for PUMA 600

Center-of-Mass and Relative Link Mass

We assume that all masses are nonzero and focus on coordinate $\mathbf{r}_i^i$ in each link i. Thus, the center-of-mass of link i in the ith coordinate frame is denoted by $\bar{\mathbf{r}}_i^i = [\bar{x}_i^i \ \bar{y}_i^i \ \bar{z}_i^i]'$ and values of center-of-mass and relative link mass are shown in Table 3.3.

Link	$\bar{x}(cm)$	$\bar{y}(cm)$	$\bar{z}(cm)$	Relative Mass(m_i/m_6)
1	0	0	8	33.5
2	−21.6	0	21.75	77.3
3	0	0	21.6	36.3
4	0	2	0	8.95
5	0	0	2	2.39
6	0	0	1	1

Table 3.3: Center-Of-Mass and Relative Link Mass

The total weight of the PUMA 500 series manipulator is 120 pounds [Unimate 85]. After subtracting the weight of base, the remaining mass of six links is assumed to be about 60 pounds. By equating the relative link mass, each link mass is shown in "PUMA_SPEC_DEF.h" in Appendix 7.

Gravitational Acceleration Vector

The gravitational acceleration vector $\mathbf{g}$ points in the negative Z_0−axis direction and it is denoted as below:

$$\mathbf{g} = \begin{bmatrix} 0 \\ 0 \\ -|g| \\ 0 \end{bmatrix}$$

where $g = 9.8062 m/sec^2$.

Pseudo-Inertia Matrix

Pseudo-inertia matrix J_i is assumed to be diagonal in [Paul 81], [Paul&R.&Z. 83]. So off-diagonal pseudo-inertias are zero and the main diagonal is composed of radii-of-gyration.

$$J_i = \begin{bmatrix} J_{ixx} & 0 & 0 & 0 \\ 0 & J_{iyy} & 0 & 0 \\ 0 & 0 & J_{izz} & 0 \\ 0 & 0 & 0 & m_i \end{bmatrix}$$

where

$$\begin{aligned} J_{ixx} &= \frac{1}{2} m_i(-S^2_{ixx} + S^2_{iyy} + S^2_{izz}) \\ J_{iyy} &= \frac{1}{2} m_i(S^2_{ixx} - S^2_{iyy} + S^2_{izz}) \\ J_{izz} &= \frac{1}{2} m_i(S^2_{ixx} + S^2_{iyy} - S^2_{izz}) \end{aligned}$$

The experimental values of radii-of-gyration for the PUMA 600 are shown in Table 3.4.

Link	$S^2_{xx}(cm^2)$	$S^2_{yy}(cm^2)$	$S^2_{zz}(cm^2)$
1	451	451	57.9
2	565.7	1847	1408
3	672.8	679.1	36
4	31.6	21.1	31.6
5	6.9	11.2	6.9
6	33.8	33.8	0.911

Table 3.4: Radii-of-Gyration

3.4 Conclusion

The customized closed-form of dynamics equation for six d-o-f PUMA robot can be realized on commercially available processes and is applicable to real-time control algorithms which require the on-line evaluation of manipulator dynamics.

C programs in Appendix 7 shows that the complete customized closed-form of dynamics equations based on L-E dynamics is realizable for real-time control applications such as having computation time less than $1.8ms$.

Chapter 4

Adaptive Self-tuning Controller

Self-tuning control is a discrete-time method which attempts to overcome problems of model-plant mismatch by automating the overall design procedure and repeating the steps of identification and controller design during each sampling period. The self-tuning controller (STC) therefore has the ability to tune the change of plant dynamics continuously. The scheme of self-tuning control is shown in Figure 5.

4.1 Time-Series Difference Model and Adaptive Self-tuning Controller

To construct an adaptive STC, a time-series difference equation model, for example, an ARX-model or an ARMAX-model may be chosen to describe the input and output relationships in the manipulator. The parameters in such a model can be computed recursively on-line at each sampling instant. The control gain is adjusted according to the updated parameter estimates and measurements so as to make the system achieve the desired goals that

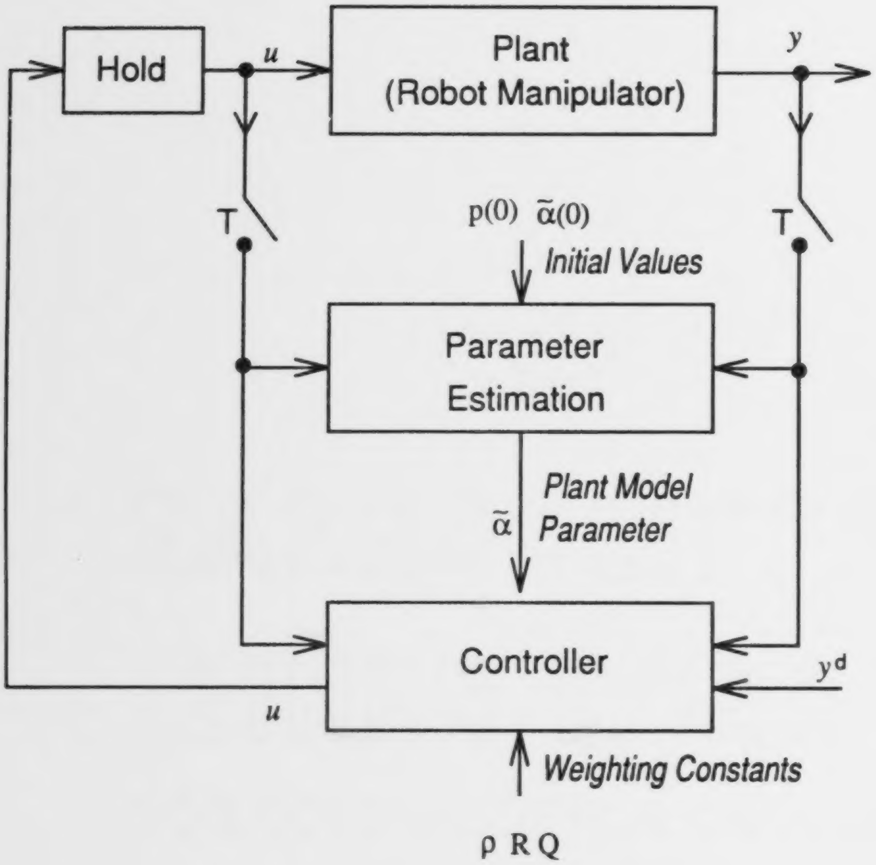
Hold
u
Plant
(Robot Manipulator)
y
T
T
p(0) α̃(0)
Initial Values
Parameter
Estimation
Plant Model
α̃
Parameter
Controller
u
yd
Weighting Constants
ρ R Q

Figure 5: Explicit Self-tuning Control Scheme

are given as the design specifications. Thus, it inherently adapts to unknown operating conditions that are conveyed by the measurements to the controller.

In applying STC to a robot manipulator, a possible approach is to linearize the dynamics model and then to discretize the linearized model with respect to time. Alternatively, the designer may choose to assume a linear discrete time series model at the beginning of the design. Then, an adaptive STC can be designed for controlling robot manipulator, where the parameters of a difference equation model are estimated and controller gain is calculated at each sampling instant using available measurements.

The independent joint control can be performed by a single-input-single-output(SISO) model when the manipulator is operating at slow or moderate speeds [Koivo&G. 83]. The interacting joint control can be designed based on a multi-input-multi-output(MIMO) model which will account for the interacting generalized forces between the joints, when the the manipulator is operating at higher speeds [Souissi&K. 87].

4.2 SISO Adaptive Self-tuning Controller for Independent Joint Dynamics

The SISO self-tuning controller is designed based on a linear model. The parameters of this model are then determined so that the best fit of the model-generated values to the measured input-output values is obtained in the sense of the sum of the least-squared errors. Then, an explicit (indirect) self-tuning LQG (Linear Quadratic Gaussian) controller is designed based on the estimated parameters of linear model [Astrom 80].

4.2.1 SISO ARX-model for Manipulator Dynamics

To design a controller under the assumption of independent joint dynamics, that is, one joint does not affect other joints, a SISO model is chosen to represent the input-output measurements of each joint. A linear discrete time-series model of autoregressive type with external excitation (ARX-model) as in equation 4.1 can be used. The number of the measured output variables is same as the number of input variables.

$$\theta_i(k) = A_i(z^{-1})\theta_i(k-1) + B_i(z^{-1})u_i(k-1) + g_i + e_i(k) \tag{4.1}$$

where $\theta_i(k)$ and $u_i(k)$ represent the measured output and input respectively of joint i at time kT; T the sampling period; g_i a possible bias due to gravitational force; $e_i(k)$ the random, zero-mean, white Gaussian noise; $A_i(z^{-1})$ and $B_i(z^{-1})$ are polynomials in the delay operator z^{-1}:

$$A_i(z^{-1}) = a_i^1 + a_i^2 z^{-1} + \cdots + a_i^n z^{-n+1} \tag{4.2}$$

$$B_i(z^{-1}) = b_i^0 + b_i^1 z^{-1} + \cdots + b_i^{n-1} z^{-n+1} \tag{4.3}$$

Once the ARX-model is constructed, the unknown parameters in equations 4.2, 4.3 and g_i can be estimated by using the measurements of input and output. The parameter estimation algorithm for SISO ARX-model will be given in the next section. The polynomial with the estimated parameters, for example, $\tilde{A}_i(z^{-1})$, $\tilde{B}_i(z^{-1})$ and $\tilde{g}_i$ are substituted into equation 4.1 for the unknown parameters, and the resulting model is used in the design of the controller.

4.2.2 Parameter Estimation in SISO ARX-model

To estimate the unknown parameters in equation 4.1, vectors α_i and ϕ_i are defined:

$$\alpha_i = [a_i^1 \cdots a_i^n \; ; \; b_i^0 \cdots b_i^{n-1} \; ; \; g_i]' \tag{4.4}$$

$$\phi_i(k-1) = [\theta_i(k-1) \cdots \theta_i(k-n); u_i(k-1) \cdots u_i(k-n); 1]' \tag{4.5}$$

where the indices i and n account for joint, $i = 1 \cdots m (m = 6$ for a six-joint manipulator) and the order of the model, respectively. Then equation 4.1 may be rewritten as follows:

$$\begin{aligned} \theta_i(k) &= \alpha_i' \phi_i(k-1) + e_i(k) \\ &= a_i^1 \theta_i(k-1) + \cdots + a_i^n \theta_i(k-n) \\ &\quad + b_i^0 u_i(k-1) + \cdots + b_i^{n-1} u_i(k-n) + e_i(k) \end{aligned} \tag{4.6}$$

From the equation 4.6, it is noted that the parameters related to ith joint only affect to the ith-joint output $\theta_i(k)$ and input $u_i(k)$. The unknown parameter vector in equation 4.6 is estimated by minimizing the following error criterion, that is,

$$H(\alpha_i) = \frac{1}{N+1} \sum_{k=n}^{N+n} \gamma_i^{N+n-k} e_i^2(k) \tag{4.7}$$

where forgetting factor γ_i, which is generally chosen to be between 0.95 and 1.0, assigns different weights on the errors depending on their relative importance. Very often, past errors are less important than present ones. $N + n$ indicates the time of the last measurement. The estimate $\tilde{\alpha}_i(k)$ of the unknown parameter vector $\alpha_i(k)$ at time kT is obtained by minimizing the function $H(\alpha_i)$

with respect to α_i in equation 4.7. The solution to the recursive least-squares problem is furnished by the following recursive equations [Ljung 83]. They constitute what is known as the *recursive least-squares*(RLS) algorithm for parameter estimation.

$$\tilde{\alpha}_i(k) = \tilde{\alpha}_i(k-1) + p_i(k)\phi_i(k-1)[\theta_i(k) - \phi_i'(k-1)\tilde{\alpha}_i(k-1)] \tag{4.8}$$

$$p_i(k) = \frac{1}{\gamma_i}\left[p_i(k-1) - \frac{p_i(k-1)\phi_i(k-1)\phi_i'(k-1)p_i(k-1)}{\gamma_i + \phi_i'(k-1)p_i(k-1)\phi_i(k-1)}\right] \tag{4.9}$$

The second term on the right side of equation 4.8 describes the correction term; $p_i(\cdot)$ is a $(2n+1)\times(2n+1)$ symmetric matrix; the initial value $\tilde{\alpha}_i(0)$ and $p_i(0)$ may be approximated. The parameter estimates will be calculated on-line using equations 4.8 and 4.9. The estimated parameters are then substituted into equation 4.1, resulting in equation 4.10.

$$\theta_i(k) = \tilde{A}_i(z^{-1})\theta_i(k-1) + \tilde{B}_i(z^{-1})u_i(k-1) + \tilde{g}_i + e_i(k) \tag{4.10}$$

The ARX-model equation 4.10 with known(or estimated) parameters is used to design an adaptive self-tuning LQG controller so that the system tracks the desired output variable $\theta_i^d(k)$.

4.2.3 Optimal LQG Controller Design for SISO system

The goal of the controller is to make the output $\theta_i(k)$ of the ith joint, $i = 1 \cdots m$, track the sequence of the desired values $\theta_i^d(k)$. The tracking of the desired output can be achieved when the control $u_i(k)$ is so determined that a chosen performance criterion $J_k(u_i)$ is minimized:

$$J_k(u_i) = E\left\{[\theta_i(k+1) - \theta_i^d(k+1)]^2 + \mu_i[u_i(k)]^2 \,/\, \Sigma_i(k)\right\} \tag{4.11}$$

where $E\{\cdot/\Sigma_i(k)\}$ denotes the expectation operation given the past values of the output and the input, that is, $\Sigma_i(k) = \{\theta_i(j), u_i(j-1), j \leq k\}$, and the weighting factor μ_i is a nonnegative number.

The performance index in equation 4.11 is minimized with respect to admissible control input $u_i(k)$ while satisfying the model equation 4.10 whose unknown parameters are replaced to the estimated parameters. The control input $u_i(k)$ at time kT is admissible when it is a function of available measurements up to and including time kT. The equation 4.10 is solved for $\theta_i(k+1)$, and resulting expression is substituted into equation 4.11 to obtain:

$$\begin{aligned} J_k(u_i) &= E\left\{[\tilde{A}_i(z^{-1})\theta_i(k) + \tilde{b}_i^0 u_i(k) + \tilde{B}_{ir}(z^{-1})u_i(k) + g_i - \theta_i^d(k+1)]^2 \right. \\ &\quad \left. + \mu_i[u_i(k)]^2 \,/\, \Sigma_i(k)\right\} \end{aligned} \tag{4.12}$$

where $\tilde{B}_{ir}(z^{-1}) = \tilde{b}_i^1 z^{-1} + \cdots + \tilde{b}_i^{n-1} z^{-n+1} = \tilde{B}_i(z^{-1}) - \tilde{b}_i^0$. The minimization of $J_k(u_i)$ in equation 4.12 with respect to $u_i(k)$ gives:

$$u_i(k) = \frac{\tilde{b}_i^0}{(\tilde{b}_i^0)^2 + \mu_i}[\theta_i^d(k+1) - \tilde{A}_i(z^{-1})\theta_i(k) - \tilde{B}_{ir}(z^{-1})u_i(k)] \tag{4.13}$$

The equation 4.13 specifies the controller that makes the output variable $\theta_i(k)$ track the desired trajectory by consuming the minimum energy of control signal $u_i(k)$ in equation 4.11. The block diagram in Figure 5 shows the overall implementation of adaptive self-tuning LQG controller.

Realization of Self-tuning LQG Controller

- **Step 1** : Select the initial values and constants of $\tilde{\alpha}_i(0), p_i(0), u_i(0), \mu_i$ for each ith joint, $i = 1, \cdots, m$(m=6 for a six-joint).
- **Step 2** : Estimate parameter vector $\tilde{\alpha}_i(k)$ by using the equations 4.8 and 4.9
- **Step 3** : Compute the control law $u_i(k)$ using the equation 4.13
- **Step 4** : Apply the computed control of $u_i(k)$ to the manipulator system for obtaining the next measurement of output $\theta_i(k+1)$.
- **Step 5** : Repeat the procedures from Step 2 to Step 4.

4.3 MIMO Self-tuning Controller for Interacting Joint Dynamics

In section 4.2, the SISO controller was designed under the assumption of the independent joint dynamics, which implies that one joint dynamics does not affect the other remaining joint dynamics. While this controller is expected to work well at the moderate operating speeds when the effects of Coriolis and centrifugal forces in manipulator dynamics are insignificant, it can be expected to exhibit difficulties in tracking when this assumption does not hold.

Therefore, in this section, the control algorithm includes interactions between the joint variables in a multi-input-multi-output(MIMO)-ARX model by using non-zero off-diagonal elements in the coefficient matrices in the model equation in the sequel.

4.3.1 MIMO ARX-model for Manipulator Dynamics

A MIMO ARX-model of in the form a difference equation can be written in the following form for a m-joint manipulator:

$$\underbrace{\Theta(k)}_{m\times 1} = \underbrace{A(z^{-1})}_{m\times m}\underbrace{\Theta(k-1)}_{m\times 1} + \underbrace{B(z^{-1})}_{m\times m}\underbrace{U(k-1)}_{m\times 1} + \underbrace{G}_{m\times 1} + \underbrace{\mathcal{E}}_{m\times 1} \tag{4.14}$$

where output vector $\Theta(k) = [\theta_1(k) \cdots \theta_m(k)]'$, input vector $U(k-1) = [u_1(k-1) \cdots u_m(k-1)]'$, gravitational vector $G = [g_1 \cdots g_m]'$ and modeling error vector $\mathcal{E} = [e_1 \cdots e_m]'$; the argument k refers to the sampling instant at time kT (the sampling period T has been omitted in the arguments); the z^{-1} is backward shift operator, that is, $z^{-1}\Theta(k) = \Theta(k-1)$; the dimensions of the vectors and matrices are as includes; The coefficient $A(z^{-1})$ and $B(z^{-1})$ are $(m \times m)$ polynomial matrices defined as:

$$\underbrace{A(z^{-1})}_{m\times m} = \underbrace{A_1}_{m\times m} + \underbrace{A_2}_{m\times m} z^{-1} + \cdots + \underbrace{A_n}_{m\times m} z^{-n+1} \tag{4.15}$$

$$\underbrace{B(z^{-1})}_{m\times m} = \underbrace{B_0}_{m\times m} + \underbrace{B_1}_{m\times m} z^{-1} + \cdots + \underbrace{B_{n-1}}_{m\times m} z^{-n+1} \tag{4.16}$$

where the unknown coefficients in polynomials, $A_1, \cdots, A_n$ and $B_0, \cdots, B_{n-1}$ have $6 \times 6 = 36$ (for a six-joint manipulator) matrix elements to be estimated in each coefficient respectively; The n is a positive integer specifying the order of model.

4.3.2 Parameter Estimation in MIMO ARX-model

To estimate the coefficient parameters in equation 4.14, the matrix α and vector $\Phi(k-1)$ are defined as:

$$\begin{aligned}\boldsymbol{\alpha} &= \left[\overbrace{\underbrace{A_1}_{m\times m},\cdots,\underbrace{A_n}_{m\times m}}^{n}\;;\;\overbrace{\underbrace{B_0}_{m\times m},\cdots,\underbrace{B_{n-1}}_{m\times m}}^{n}\;;\;\overbrace{\underbrace{G}_{m\times 1}}^{1}\right]'_{(2mn+1)\times m} \\ &= [\,\alpha_1,\cdots,\alpha_m\,] \qquad (4.17)\end{aligned}$$

where prime $'$ denotes the transposition, and α_i is a column vector having dimension of $[\,(2mn+1)\times 1\,]$ for $i = 1\cdots m(=6$ for a six-joint manipulator) and n is the order of model, that is:

$$\begin{aligned}\alpha_i &= \Big[a_{i1}^1\cdots a_{im}^1, a_{i1}^2\cdots a_{im}^2, \bullet\bullet\bullet, a_{i1}^n\cdots a_{im}^n\;; \\ &\qquad b_{i1}^0\cdots b_{im}^0, b_{i1}^1\cdots b_{im}^1, \bullet\bullet\bullet, b_{i1}^{n-1}\cdots b_{im}^{n-1}\;;\; g_i\Big]'_{(2mn+1)\times 1} \qquad (4.18)\end{aligned}$$

$$\begin{aligned}\underbrace{\Phi(k-1)}_{(2mn+1)\times 1} &= \left[\overbrace{\underbrace{\Theta'(k-1)}_{1\times m},\bullet\bullet\bullet,\underbrace{\Theta'(k-n)}_{1\times m}}^{n}\;;\;\overbrace{\underbrace{U'(k-1)}_{1\times m},\bullet\bullet\bullet,\underbrace{U'(k-n)}_{1\times m}}^{n}\;;\;\overbrace{\underbrace{1}_{1\times 1}}^{1}\right]' \\ &= [\,\theta_1(k-1)\cdots\theta_m(k-1),\bullet\bullet\bullet,\theta_1(k-n)\cdots\theta_m(k-n)\;; \\ &\qquad u_1(k-1)\cdots u_m(k-1),\bullet\bullet\bullet,u_1(k-n)\cdots u_m(k-n)\;;\;1\,]' \\ &\qquad\qquad (4.19)\end{aligned}$$

By using the equations 4.17 and 4.19, the equation 4.14 can be rewritten as follows:

$$\underbrace{\Theta(k)}_{m\times 1} = \underbrace{\boldsymbol{\alpha}'}_{m\times(2mn+1)}\;\underbrace{\Phi(k-1)}_{(2mn+1)\times 1} + \underbrace{\mathcal{E}}_{m\times 1} \qquad (4.20)$$

where $\Theta(k) = [\theta_1(k)\cdots\theta_m(k)]'$ is a $m-$dimensional vector. Therefore, the ith joint output can be represented as follows:

$$\begin{aligned}
\theta_i(k) &= \alpha_i'\Phi(k-1) + e_i(k) \\
&= a_{i1}^{1}\theta_1(k-1) + \cdots + a_{im}^{1}\theta_m(k-1) + \bullet\bullet\bullet + a_{i1}^{n}\theta_1(k-n) + \cdots \\
&\quad + \theta_{im}^{n}\theta_m(k-n) + b_{i1}^{0}u_1(k-1) + \cdots + b_{im}^{0}u_m(k-1) + \bullet\bullet\bullet \\
&\quad + b_{i1}^{n-1}u_1(k-n) + \cdots + b_{im}^{n-1}u_m(k-n) + g_i + e_i(k) \qquad (4.21)
\end{aligned}$$

We should note in equation 4.21 that present output $\theta_i(k)$ for the ith joint is affected by past all-joint outputs $\theta_i(k-1)\cdots\theta_i(k-n)$ and inputs $u_i(k-1)\cdots u_i(k-n)$ for $i = 1\cdots m$ (m=6 for a six-joint manipulator). From this point of view, the equation 4.14 for the MIMO model is clearly different from the equation 4.6 for the SISO model.

The numerical values of the components of the unknown parameter α in the MIMO model are calculated by the recursive least-squares(RLS) error estimation method by estimating one vector $\alpha_i(k)$ at a time. The error criterion to be minimized is chosen for each vector α_i of α as follows:

$$J(\alpha_i) = \frac{1}{N+1}\sum_{k=n}^{N+n}\gamma_i^{N+n-k}e_i^2(k) \qquad (4.22)$$

where $N+n$ represents the last number of the measurements; the weighting term γ_i is called a *forgetting factor* which assigns different weight depending on their relative importance, for example, past errors are usually less important than present ones.

The problem is to minimize the error criterion $J(\alpha_i)$ with respect to vector α_i while satisfying the equation 4.21. The solution to the least-squares problem is furnished by following recursive equations:

$$\tilde{\alpha}_i(k) = \tilde{\alpha}_i(k-1) + p(k)\Phi(k-1)[\theta_i(k) - \tilde{\alpha}_i'(k-1)\Phi(k-1)] \qquad (4.23)$$

$$p(k) = \frac{1}{\gamma}\left[p(k-1) - \frac{p(k-1)\Phi(k-1)\Phi'(k-1)p(k-1)}{\gamma + \Phi'(k-1)p(k-1)\Phi(k-1)}\right] \tag{4.24}$$

where $p(\cdot)$ is a $(2mn+1) \times (2mn+1)$ symmetric matrix (recall $p_i(\cdot)$ is a $(2n+1)\times(2n+1)$ for SISO model of equation 4.9 in section 4.2.2.); the estimate of α_i at time k is written as $\tilde{\alpha}_i(k)$. Thus the parameter estimation of the multivariable time series model can be carried out on-line using equations 4.23 and 4.24. It is noted that co-variance matrix $p(\cdot)$ is of no subscript i in equation 4.24 because each estimate of $\tilde{\alpha}_i(k)$ for $i = 1 \cdots m$ holds one co-variance matrix in common.

4.3.3 Optimal LQG Controller Design for MIMO System

The self-tuning LQG controller is constructed by minimizing the following quadratic cost-function:

$$J_k(U) = E\left\{\, \|\Theta^m(k+1) - \Theta^d(k+1)\|_Q^2 + \|U(k)\|_R^2 \;/\; \Sigma(k) \,\right\} \tag{4.25}$$

where $\|\cdot\|_R^2$ indicates the generalized norm with weight R, for example, $\|U\|_R^2 = U'RU$. Here, cost-function weighting matrices R and Q are the major design parameters which must be selected by the controller designer; $\Theta^m(k+1)$ and $\Theta^d(k+1)$ describe the measured and desired trajectory vectors as a sequence of discrete point respectively. The expectation operation is conditioned on the available measurements up to and including time kT.

The problem is to minimize the cost-function with respect to the admissible controls while satisfying the following constraint equation:

$$\Theta^m(k) = \tilde{A}(z^{-1})\Theta^m(k-1) + \tilde{B}(z^{-1})U(k-1) + \tilde{G} + \mathcal{E} \qquad (4.26)$$

The unknown parameters $\tilde{A}$, $\tilde{B}$, and $\tilde{G}$ can be obtained by recursive parameter estimation equations 4.23 and 4.24. The equation 4.26 can be rewritten as an one sampling period advanced form as follows:

$$\Theta^m(k+1) = \tilde{A}(z^{-1})\Theta^m(k) + \tilde{B}(z^{-1})U(k) + \tilde{G} + \mathcal{E} \qquad (4.27)$$

The solution for the controller is obtained by substituting equation 4.27 into the cost-function of the equation 4.25 and by minimizing the resulting equation with respect to controller gain $U(k)$. The resulting controller is expressed as follows:

$$U(k) = \left[R + \tilde{B}_0' Q \tilde{B}_0\right]^{-1} \tilde{B}_0' Q \left[\Theta^d(k+1) - \tilde{A}(z^{-1})\Theta^m(k) - \tilde{B}_r(z^{-1})U(k) - \tilde{G}\right] \qquad (4.28)$$

where

$$\begin{aligned} \tilde{A}(z^{-1}) &= \tilde{A}_1 + \tilde{A}_2 z^{-1} + \cdots + \tilde{A}_n z^{-n+1} \\ \tilde{B}_r(z^{-1}) &= \tilde{B}_1 z^{-1} + \tilde{B}_2 z^{-2} + \cdots + \tilde{B}_{n-1} z^{-n+1} \end{aligned}$$

4.4 MIMO Self-tuning LQG Controller for Controlling Velocity Variables

In the previous sections, the measured output is the joint position, i.e., $\theta(k)$. When a self-tuning controller is designed based on $\theta(k)$, difficulties in tracking may be experienced when the ARX-model for the input and output of joint position is of low order, and/or the noise process deviates considerately from the Gaussian assumption.

The aforementioned problems may often be circumvented by making joint velocity as the output variable of the time series model. These joint velocities can directly be measured by tachometer or calculated from two adjacent positional readings. The time-series MIMO ARX-model becomes:

$$\dot{\Theta}^m(k) = A(z^{-1})\dot{\Theta}^m(k-1) + B(z^{-1})U(k-1) + G + \mathcal{E} \tag{4.29}$$

where $\dot{\Theta}^m(k) = [\dot{\theta}_1^m(k)\,\dot{\theta}_2^m(k)\,\cdots\,\dot{\theta}_m^m(k)]'$ represents the measured joint angular velocity vector; $U(k-1) = [u_1(k-1)u_2(k-1)\cdots u_m(k-1)]'$ accounts for the voltages applied to the motors of the joints at time $(k-1)T$; $G = [g_1\ g_2\ \cdots\ g_m]'$ is a m-dimensional possible biases, i.e., gravitational forces; $\mathcal{E} = [e_1\ e_2\ \cdots\ e_m]'$ is a m-dimensional equation error vector, where each $e_i(k)$, $i = 1\cdots m$, represents a white Gaussian zero-mean noise process, which is independent of current and past outputs and inputs.

For performing the parameter estimates in equation 4.29, similar definitions and procedures as in section 4.3.2 are used as before.

$$\begin{aligned} \boldsymbol{\alpha} &= [A_1,\cdots,A_n\ ;\ B_0,\cdots,B_{n-1}\ ;\ G]' \\ &= [\alpha_1,\cdots,\alpha_m] \end{aligned} \tag{4.30}$$

$$\Phi(k-1) = [\ \dot{\Theta}^{m\prime}(k-1),\ldots,\dot{\Theta}^{m\prime}(k-n)\ ;\ U'(k-1),\cdots,U'(k-n)\ ;\ 1\ ]' \tag{4.31}$$

$$\tilde{\alpha}_i(k) = \tilde{\alpha}_i(k-1) + p(k)\Phi(k-1)[\dot{\theta}_i(k) - \tilde{\alpha}_i'(k-1)\Phi(k-1)]' \tag{4.32}$$

$$p(k) = \frac{1}{\gamma}\left[p(k-1) - \frac{p(k-1)\Phi(k-1)\Phi'(k-1)p(k-1)}{\gamma + \Phi'(k-1)p(k-1)\Phi(k-1)}\right] \tag{4.33}$$

The LQG controller with self-tuning is determined by minimizing the position and velocity squared-errors and the energy as specified by the cost-function as follows:

$$\begin{aligned} J_k(U) &= E\left\{\|[\dot{\Theta}^m(k+1) - \dot{\Theta}^d(k+1)] + \rho[\Theta^m(k+1) - \Theta^d(k+1)]\|_Q^2 \right. \\ &\quad \left. + \|U(k)\|_R^2/\Sigma(k)\right\} \end{aligned} \tag{4.34}$$

where superscripts m and d indicate the measured and desired signals respectively; desired position $\Theta^d(k+1)$ is approximately represented by $\Theta^d(k+1) = \Theta^d(k) + \dot{\Theta}^d(k)T$ because $\Theta^d(k+1)$ is unknown at time kT, and prediction algorithms may be used for the desired signals, for example, $\dot{\Theta}^d(k+1)$ is predicted one sampling period early in real applications of vision feedback servoing [Allen&Y.&T. 90], [Koivo&H. 91].

The cost-function weighting matrices $\rho = \rho' = diag[\rho_1 \cdots \rho_m] \geq 0$, $R = R' \geq 0$, and $Q = Q' > 0$ are chosen in view of the specific operational objects. ρ emphasizes the relative importance of position tracking errors with respect to velocity tracking errors.

The control law, determined by minimizing the equation 4.34 subject to constraint equation 4.29 with estimates computed by equations 4.30 - 4.33, is shown in below.

$$\begin{aligned} U(k) &= \left[R + \tilde{B}_0'Q\tilde{B}_0\right]^{-1} \tilde{B}'Q\left\{\dot{\Theta}^d(k+1) - \tilde{A}(z^{-1})\dot{\Theta}^m(k) - \tilde{B}_r(z^{-1})U(k)\right. \\ &\quad \left. - \tilde{G} + \rho\left[\Theta^d(k+1) - \Theta^m(k+1)\right]\right\} \end{aligned} \tag{4.35}$$

where tilde ~ represents the estimates; the $(m \times m)$ matrices A and B are polynomials defined by:

$$\tilde{A}(z^{-1}) = \tilde{A}_1 + \tilde{A}_2 Z^{-1} + \cdots + \tilde{A}_n Z^{-n+1} \tag{4.36}$$

$$\tilde{B}_r(z^{-1}) = \tilde{B}_1 Z^{-1} + \tilde{B}_2 Z^{-2} + \cdots + \tilde{B}_{n-1} Z^{-n+1} \tag{4.37}$$

The MIMO ARX-model having control-variables of velocities in equation 4.29 is of the same form as the MIMO ARX-model having control-variables of positions in equation 4.26. However, the two control laws (equation 4.28 for the position controller) and (4.35 for the velocity controller) are different in terms of their capability to compensate for tracking errors. The velocity controller can control both positions and velocities, while position controller can control only the positions.

Chapter 5

Computer Simulations and Development

In this Chapter, simulation studies carried out to test the performance of the of the controllers designed in Chapter 4 are presented. The results are presented in four parts.

1. Digital Simulation of PUMA 600 Motion;
2. Transformation of the multi-input-multi-output (MIMO) STC a single-input-single-output (SIMO) STC;
3. Testing the proposed SIMO STC;
    - Test1: At moderate high velocity without payload.
    - Test2: At very high velocity with payload.
4. SIMO STC with weakened couplings/non-linearities;
    - Test3: At very high velocity with payload.

Since all computations in this thesis are performed by matrix manipulations, i.e., matrix augmentations, inversions, multiplications, subtractions, additions, multiplications by a scalar or different scalars etc., the main program is built up by calling the matrix tool functions written in C program language. Tool-box for controlling and simulating the robot manipulator has been recently presented by using the MATLAB instead of C language [Honey&J. 92]. In this thesis, dynamics equations given in Appendix 7 may be useful in the future robotics-related works for simulating the motion of all revolute six-joint robot manipulator in both forward and inverse dynamics.

5.1 Digital Simulation of manipulator Motion

Simulation of physical manipulator motion implies forward dynamics matter, that is, given certain torques/forces, the joint acceleration is solved. This requires to solve the inverse dynamics equation 3.29 as below:

$$\ddot{\Theta}(k) = \mathbf{D}^{-1}(\Theta(k)) \left[\boldsymbol{\tau}(k) - \mathbf{C}(\Theta(k), \dot{\Theta}(k)) - \mathbf{G}(\Theta(k))\right] \tag{5.1}$$

For the recursive calculation of 5.1 at each time k, the initial values of $\boldsymbol{\tau}(0)$, $\Theta(0)$ and $\dot{\Theta}(0)$ should be given by controller.

The kinematic structural parameters to compute $\mathbf{D}(\cdot)$, $\mathbf{C}(\cdot)$ and $\mathbf{G}(\cdot)$ in equation 5.1 are given in Table 2.1. The inversion of a inertia matrix, $\mathbf{D}(\cdot)$, is done by using the well known lower/upper triangular(LU) decomposition and backsubstitution. For some applications on singular matrices, LU decomposition method substitutes tiny value(1.0e-20) for zero pivot element [Press&F.&T.&V. 88].

5.1.1 Measured Angular Velocity and Position with Measurement Error

Velocity and position are computed from acceleration equation 5.1.

$$\dot{\Theta}(k+1) = \dot{\Theta}(k) + \ddot{\Theta}(k)T_s \tag{5.2}$$

$$\Theta(k+1) = \Theta(k) + \dot{\Theta}(k)T_s + \frac{1}{2}\ddot{\Theta}(k)T_s^2 \tag{5.3}$$

The sampling period $T_s = 0.01$ second is used in the sense that breaking continuous times into increments would be a reasonable approximation. The measurements of joint velocity $\dot{\Theta}^m(k)$ is generated by superimposing sample values of white Gaussian noise process with zero-mean and variance σ^2, which accounts for measurement error.

$$\dot{\Theta}^m(k) = \dot{\Theta}(k) + \mathcal{N}(k) \tag{5.4}$$

where $\dot{\Theta}^m(k)$ is a (6×1) measured velocity vector, $\dot{\Theta}(k)$ is a (6×1) pure velocity vector from inverse dynamics equations and $\mathcal{N}(k)$ is a (6×1) measurement error vector whose elements have independent samples of white Gaussian noise process with $\sigma^2 = 0.01$.

Here, only measurements of joint velocity are assumed to be available because the measurement of joint position can be computed by integrating the joint velocity measurements of equation 5.4 in the controller unit:

$$\begin{aligned}
\Theta^m(k) &= \Theta^m(k-1) + \dot{\Theta}^m(k-1)T_s + \frac{1}{2}\ddot{\Theta}^m(k-1)T_s^2 \\
&= \Theta^m(k-1) + \dot{\Theta}^m(k-1)T_s + \frac{1}{2}\left[\frac{\dot{\Theta}^m(k) - \dot{\Theta}^m(k-1)}{T_s}\right]T_s^2 \\
&= \Theta^m(k-1) + \frac{1}{2}\dot{\Theta}^m(k-1)T_s + \frac{1}{2}\dot{\Theta}^m(k)T_s
\end{aligned} \tag{5.5}$$

where $\Theta^m(k-1)$, $\dot{\Theta}^m(k-1)$ and $\dot{\Theta}^m(k)$ are all available at instance k.

5.1.2 About Generation of Gaussian Noise

The white Gaussian zero-mean process with variance σ^2 is generated with random numbers from random number generating function in the program. However, random number generating function requires different "seeds" as an argument for generating different pattern of numbers in each execution of program. Generally this value of seed is fed into program manually. In this thesis, different seeds are automatically generated into program by reading the present time from SUN computer clock and then multiplied hours, minutes and seconds by different weights as follows:

$$\text{seed} = (hours \times 10000) + (minutes \times 100) + seconds \qquad (5.6)$$

where hours, minutes and seconds are present time read by time-reading function in program. With these different seeds, each execution of main program uses different measurement error patterns of Gaussian noise process.

5.2 Design of Adaptive STC on SIMO ARX-model

Designer should remind that the practical tips in adaptive STC is to reduce the number of model parameters to be estimated to minimum, and to give these parameters a clear physical interpretation.

5.2.1 Transformation MIMO ARX-model to SIMO ARX-model

In order to decrease the number of parameters, the MIMO ARX-model is converted into SIMO ARX-model. The procedures for doing this is as follows:

Since the external input $u_i(k-1)$ is assumed not to interact with other inputs $u_j(k-1)$ for $i, j = 1 \cdots 6$ and $i \neq j$, coefficient matrix for inputs B_0 should be assumed to be a diagonal matrix which converts the MIMO-

ARX model into single-input-multi-output(SIMO) ARX-model. Thereby, the number of elements to be estimated is reduced from 36 to 6.

If the above assumption is not reasonable, either there will be a large tracking error or the system may even go out of control. It should be noted however that the A_1 for six joint velocities is a full (6×6) matrix because the joint-velocities are expected to interact with each other.

$$\underbrace{\dot{\Theta}(k)}_{6\times 1} = \underbrace{A_1}_{6\times 6} \underbrace{\dot{\Theta}(k-1)}_{6\times 1} + \underbrace{B_0}_{6\times 6} \underbrace{U(k-1)}_{6\times 1} + \underbrace{G}_{6\times 1} + \underbrace{\mathcal{E}}_{6\times 1} \tag{5.7}$$

where

$$A_1 = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{16} \\ a_{21} & a_{22} & \cdots & a_{26} \\ \vdots & \vdots & \ddots & \vdots \\ a_{61} & a_{62} & \cdots & a_{66} \end{bmatrix} \tag{5.8}$$

$$B_0 = \begin{bmatrix} b_{11} & 0 & 0 & 0 & 0 & 0 \\ 0 & b_{22} & 0 & 0 & 0 & 0 \\ 0 & 0 & b_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & b_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & b_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & b_{66} \end{bmatrix} \tag{5.9}$$

$$G = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \\ g_6 \end{bmatrix} \tag{5.10}$$

$$\mathcal{E} = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \end{bmatrix} \tag{5.11}$$

where G vector accounts for gravitational forces; $\mathcal{E}$ is a modeling error vector. Hence, at least $48 = 36 + 6 + 6$ parameter elements have to be estimated if the coupling effects are taken into consideration.

5.2.2 Parameter Estimation in SIMO ARX-model By RLS

In order to estimate parameters in SIMO ARX-model at each sampling period, the *regression vectors* $\Phi_i(k-1), i = 1 \cdots 6$ and *parameter matrix* $\alpha(k-1)$ are defined by:

$$\begin{aligned} \Phi_1(k-1) &= \left[\dot{\theta}_1(k-1) \cdots \dot{\theta}_6(k-1)\ ;\ u_1(k-1)\ ;\ 1\right]' \\ &\ \vdots \\ \Phi_6(k-1) &= \left[\dot{\theta}_1(k-1) \cdots \dot{\theta}_6(k-1)\ ;\ u_6(k-1)\ ;\ 1\right]' \end{aligned} \tag{5.12}$$

$$\alpha(k-1) = [A_1\ ;\ \mathcal{B}_0\ ;\ G]'$$

$$= \begin{bmatrix} a_{11} & \cdots & a_{16} & ; & b_{11} & ; & g_1 \\ a_{21} & \cdots & a_{26} & ; & b_{22} & ; & g_2 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ a_{61} & \cdots & a_{66} & ; & b_{66} & ; & g_6 \end{bmatrix}'$$

$$= [\alpha_1 \quad \alpha_2 \quad \cdots \quad \alpha_6] \qquad (5.13)$$

where $\mathcal{B}_0$ and α_i are column vectors as below:

$$\mathcal{B}_0 = \begin{bmatrix} b_{11} \\ b_{22} \\ \vdots \\ b_{66} \end{bmatrix}$$

$$\alpha_1 = \begin{bmatrix} a_{11} \\ \vdots \\ a_{16} \\ b_{11} \\ g_1 \end{bmatrix}$$

$$\vdots$$

$$\alpha_6 = \begin{bmatrix} a_{61} \\ \vdots \\ a_{66} \\ b_{66} \\ g_6 \end{bmatrix}$$

Plant model equation 5.7 is rewritten in the estimation form of SIMO ARX-model:

$$\begin{aligned}\underbrace{\dot{\theta}_i(k)}_{1\times1} &= \underbrace{\alpha_i'(k-1)}_{1\times8}\underbrace{\Phi_i(k-1)}_{8\times1} + \underbrace{e_i}_{1\times1} \\ &\text{for} \quad i = 1\cdots6\end{aligned} \tag{5.14}$$

where output variable $\dot{\theta}_i$ represents a measured joint velocity in each joint. Equation 5.14 implies interactions between joint velocity variables. Then, parameter estimation algorithm by recursive least-squares(RLS) for SIMO ARX-model is presented in equation 5.15 and 5.16, where each $\alpha_i(k)$ uses its own co-variance matrix $p_i(k)$, which is different from RLS algorithm in MIMO ARX-model in Chapter 4.

$$\tilde{\alpha}_i(k) = \tilde{\alpha}_i(k-1) + p_i(k)\Phi_i(k-1)[\dot{\theta}_i(k) - \tilde{\alpha}_i'(k-1)\Phi_i(k-1)]' \tag{5.15}$$

$$p_i(k) = \frac{1}{\gamma}\left[p_i(k-1) - \frac{p_i(k-1)\Phi_i(k-1)\Phi_i'(k-1)p_i(k-1)}{\gamma + \Phi_i'(k-1)p_i(k-1)\Phi_i(k-1)}\right] \tag{5.16}$$

where γ is known as the *forgetting factor* and may be slightly less than one(in this thesis, $\gamma = 0.99$). Choice of the value of γ is a trade-off between tracking ability and noise sensitivity: a smaller value of γ gives fast tracking but leads to a greater noise sensitivity, and the opposite is true for γ close to one.

Then, optimal LQG controller is determined based on same cost-function as MIMO ARX-model of velocity control-variable.

$$\begin{aligned}U(k) &= \left[R + \tilde{B}_0' Q \tilde{B}_0\right]^{-1} \tilde{B}_0' Q \left\{\dot{\Theta}^d(k+1) - A_1\dot{\Theta}^m(k) - \tilde{G}\right. \\ &\quad \left. + \rho\left[\Theta^d(k+1) - \Theta^m(k+1)\right]\right\}\end{aligned} \tag{5.17}$$

where the desired(referential) angular velocity $\dot{\Theta}^d(k+1)$ and position $\Theta^d(k+1)$ at one sampling period ahead are assumed to be available at insatance kT_s from the known desired trajectory of object. In the real applications, it is generally performed by the prediction algorithm [Allen&Y.&T. 90].

In the optimal LQG controller gain formula of equation 5.17, cost-function weighting matrix, Q, ρ and R, which are chosen intuitionally by designer, can be crucial factors for a adaptive STC. In the simuation studies reported here, they are selected as:

$$Q = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3.5 \end{bmatrix} \tag{5.18}$$

$$\rho = \begin{bmatrix} 5.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 6.5 \end{bmatrix} \tag{5.19}$$

$$R = \tilde{B}_0' \tilde{B}_0' \tag{5.20}$$

where $\tilde{B}_0$ is an estimate for B_0.

5.3 Simulation Results on Adaptive SIMO STC

Two typical simulation results are presented to demonstrate the effectiveness or otherwise of the proposed adaptive controller. All six angular joint positions/velocities are controlled simultaneously and tested on all revolute six-joint dynamics for a plant.

5.3.1 Tracking High Velocity and Position without Payload

To track high desired joint velocities and thereby to track the desired joint position by proposed SIMO adaptive self-tuning LQG control are successfully shown in Figure 6 - 23 when the end-effector does not pick up payload.

In Figures in each page, top graph shows the process of velocity tracking by both the desired and measured velocity profiles with respect to time X-axis, middle graph represents the joint position tracking result by desired and measured angular position profiles, which result from velocity tracking process in top graph, and bottom graph indicates joint position tracking error, that is, difference between measured and desired joint position values. These results come from by applying the adaptive SIMO STC to the plant of full revolute six-joint dynamics.

Since revolute robot manipulator needs to have at least six joints for minimum role that is to catch up an object by the end-effector in work space, controller test in simulation should be performed on the dynamics of a six-joint robot manipulator in order to expose the objective capability of controller or problems from the situation when a real robot is controlled. Since good tracking results are shown in Figure 6 - 23, the assumptions at the stage of controller design - joint velocity is coupled, while input to each joint actuator is not interacting - can be said to be reasonable.

In case of controlling velocity variable for tracking desired joint velocity and position, the resulting measured position tracking error can become smaller than position error from directly controlling the position variable because the integration of the measured velocity for getting measured position may compensate the noises or measurement errors of white Gaussian *zero*-mean noise around nominal signal. This fact may come true by taking look at both top and middle graphs in each Figure page with reminding that the proposed control is a velocity controller, thereby a position controller.

For a Test1 in this section, the desired velocity profiles are assumed to be sinusoidal *sine* curves with 114.59(deg./sec.) or 2(rad./sec.) of amplitude and 0.008(cycles/sec) of frequency for all six joints, for an example, $\theta_1^d(k) = Initial\_Velocity + Vmax * sin(2 * 3.14 * Freq * Ts * k)$. The 114.59 (deg./sec) is a kind of high joint velocity.

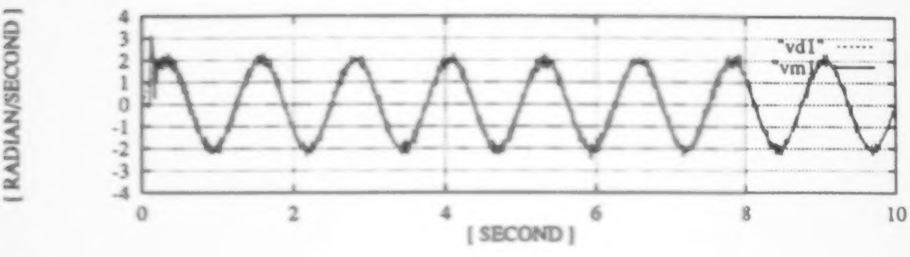

Figure 6: Test1:Velocity Tracking in Joint1 without payload

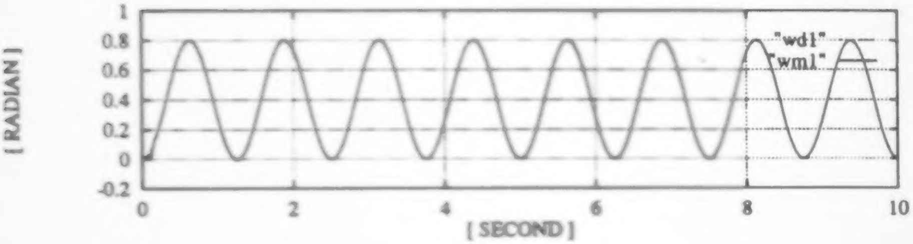

Figure 7: Test1:Position Tracking in Joint1 without payload

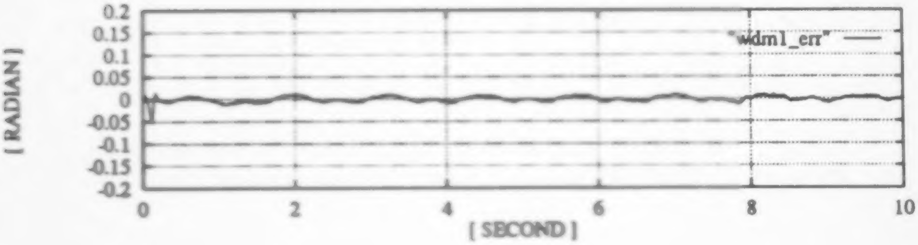

Figure 8: Test1:Position Tracking Error in Joint1 without payload

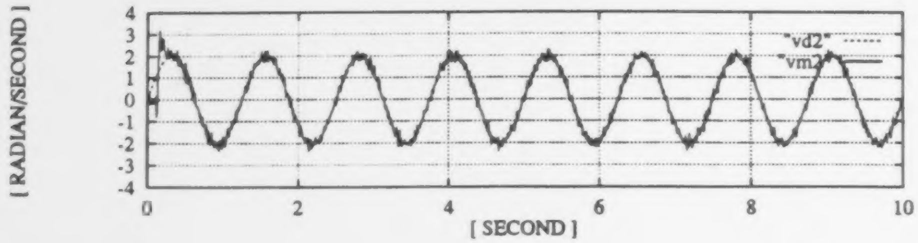

Figure 9: Test1:Velocity Tracking in Joint2 without payload

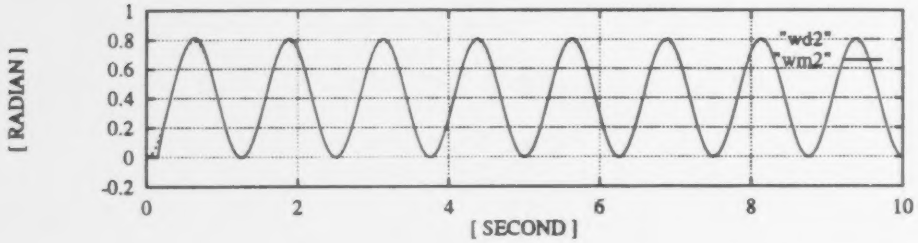

Figure 10: Test1:Position Tracking in Joint2 without payload

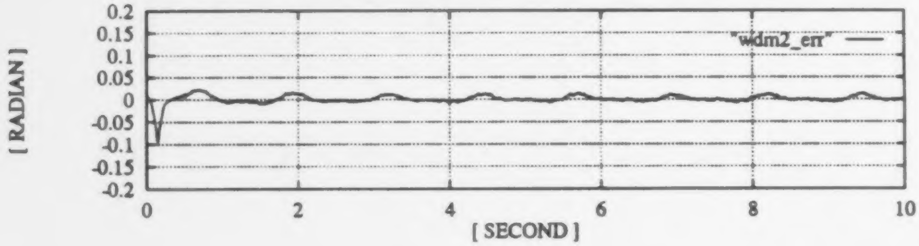

Figure 11: Test1:Position Tracking Error in Joint2 without payload

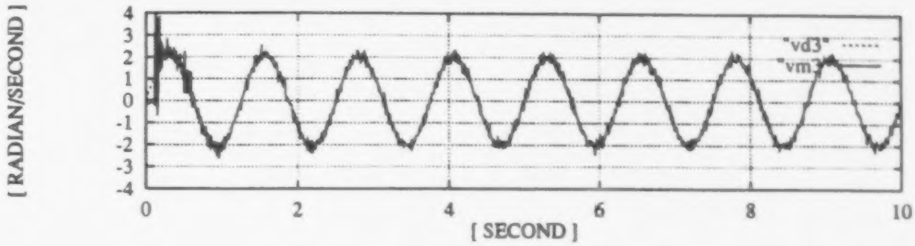

Figure 12: Test1:Velocity Tracking in Joint3 without payload

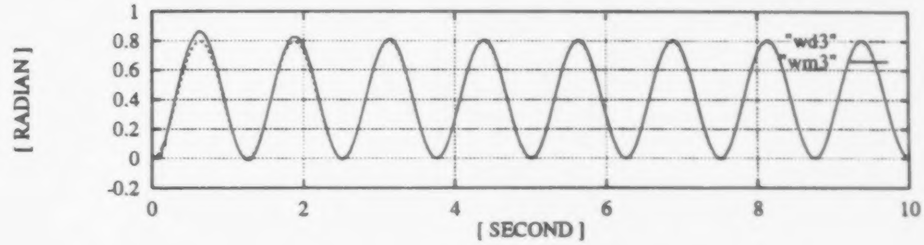

Figure 13: Test1:Position Tracking in Joint3 without payload

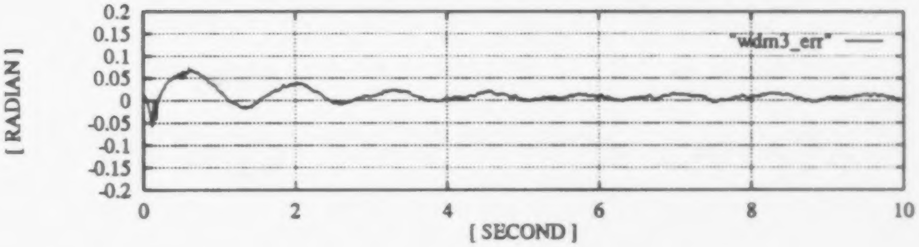

Figure 14: Test1:Position Tracking Error in Joint3 without payload

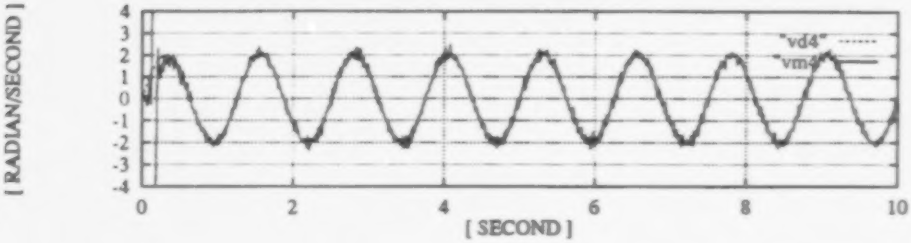

Figure 15: Test1:Velocity Tracking in Joint4 without payload

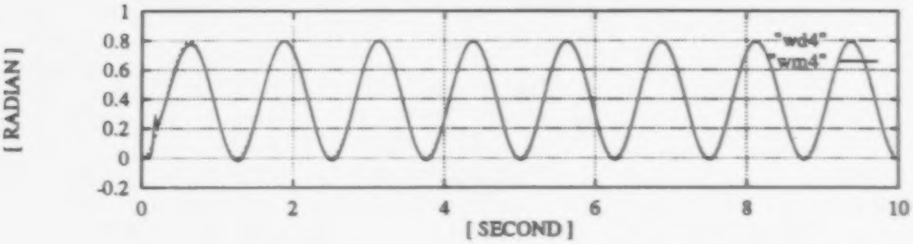

Figure 16: Test1:Position Tracking in Joint4 without payload

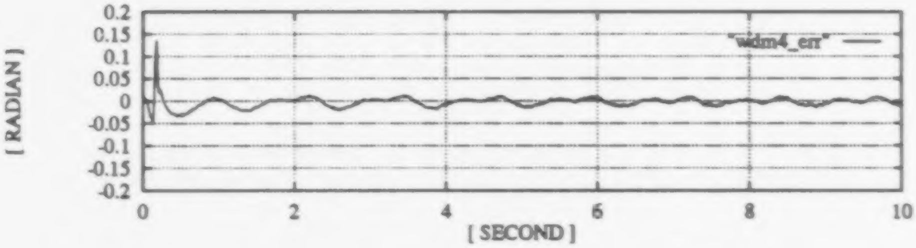

Figure 17: Test1:Position Tracking Error in Joint4 without payload

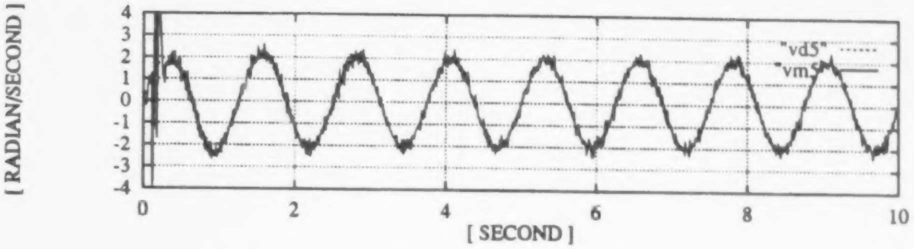

Figure 18: Test1:Velocity Tracking in Joint5 without payload

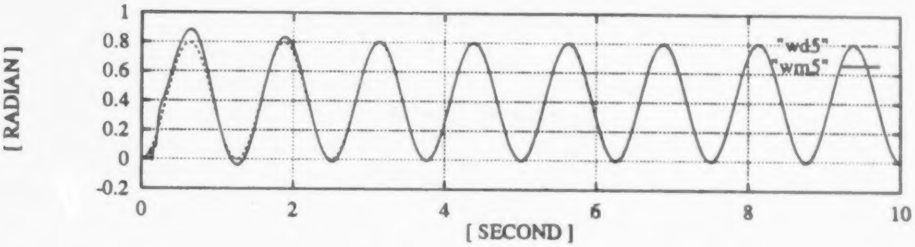

Figure 19: Test1:Position Tracking in Joint5 without payload

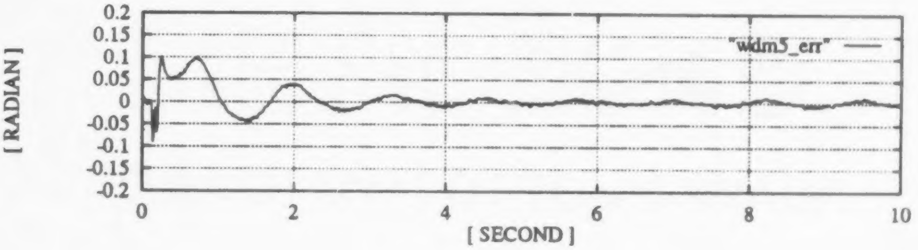

Figure 20: Test1:Position Tracking Error in Joint5 without payload

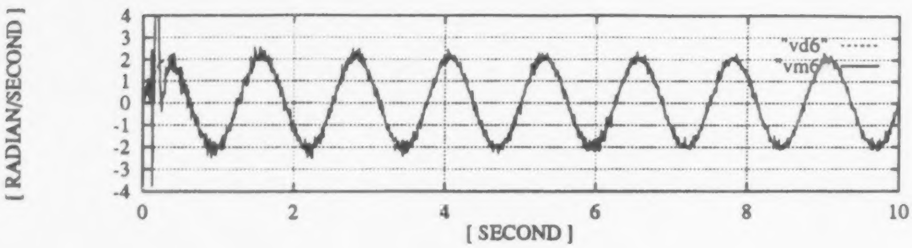

Figure 21: Test1:Velocity Tracking in Joint6 without payload

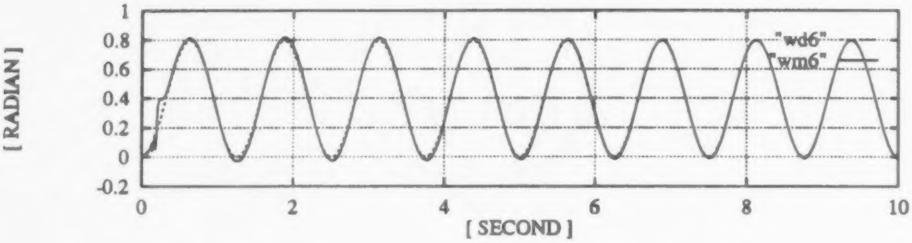

Figure 22: Test1:Position Tracking in Joint6 without payload

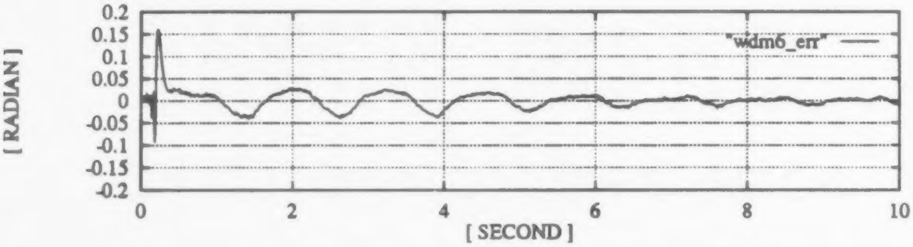

Figure 23: Test1:Position Tracking Error in Joint6 without payload

5.3.2 Tracking Very High Velocity and Position with Picking up Payload

For a Test2 in this section, the maximum desired velocity is increased from 114.59 (deg./sec.) to 200.54 (deg./sec.) or 3.5 (rad./sec.) and the end-effector picks up $1Kg$ weight of payload which is assumed to be added to the mass center of link 6 of the manipulator, and to be picked up by the end-effector after 6 seconds. But manipulator controller has no previous information about payload, i.e., weight and time of picking up. And then, same adaptive SIMO STC is used for tracking the changed position/velocity profile with payload.

In controlling the robot manipulator, 200.54(deg./sec.) is very high maximum velocity as a desired velocity profile. Since high velocity makes the dynamics of the manipulator become more nonlinear, adaptive SIMO STC based on linear system model may produce large modeling error between linear ARX-model and real plant dynamics even though couplings between joints are taken into account. Above all, the effect of loading becomes very significant at high velocity of the manipulator with payload. These changes in dynamics at high velocity make adaptive STC be very poor in tracking errors.

The simulation result with the unacceptable tracking errors in Test2 are shown in Figure 24 only for the Joint 1. Bottom graph shows an unacceptable angular position error. Especially, three graphs of velocity/position trackings and position error show the serious deviation from the desired values owing to payload. Through Test2, adaptive SIMO STC is not suitable for a manipulator which is moving at very high velocity and which is expected to pick up payload in the middle of high speed excursion.

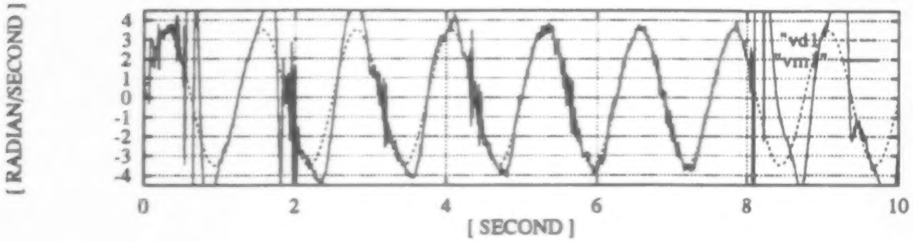

Figure 24: Test2:Velocity Tracking in Joint1 with payload

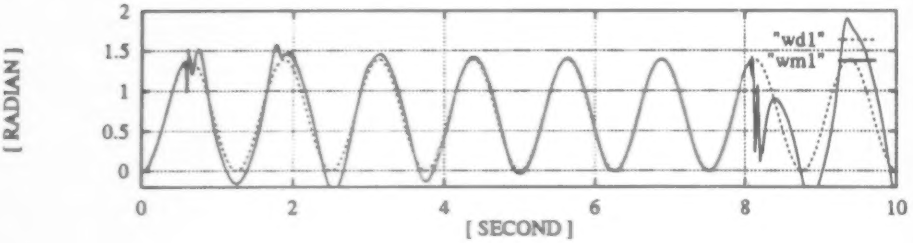

Figure 25: Test2:Position Tracking in Joint1 with payload

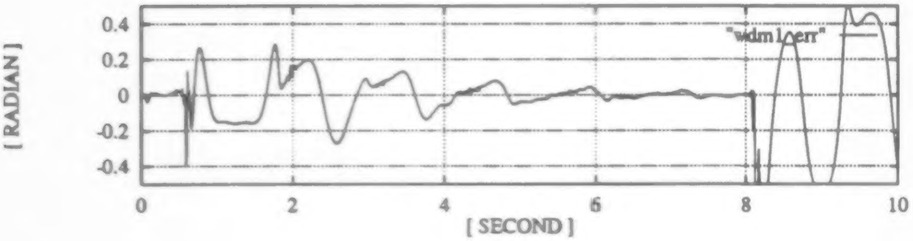

Figure 26: Test2:Position Tracking Error in Joint1 with payload

5.4 Pseudo-Linearization by Weakening Effects of Couplings and Non-linearities

We have seen that adaptive STC is not working properly with very high desired velocity profile having 3.5 (rad./sec.) maximum velocity with $1Kg$ payload in Test2, while it is working well with given desired velocity profile having 2 (rad./sec.) maximum velocity without payload in Test1.

The reasons may be conjectured from the scheme of the explicit adaptive STC in Figure 5 where there is no explicit scheme to remove the mismatch between linear system model on which adaptive STC was designed and non-linear dynamics of a real robot manipulator. The mismatch may be increased by Coriolis and centrifugal force especially at high velocity.

In order to make the dynamics of robot manipulator be close to a linear ARX-model on which adaptive STC is designed, it is necessary to compensate for the Coriolis and centrifugal force with respect to the desired trajectory by using customized L-E dynamics equation so that when the computed Coriolis and centrifugal force is subtracted from the dynamics, it is in the vicinity of linear. So, scheme of the explicit adaptive STC for controlling a robot manipulator in Figure 27 shows how the Coriolis/centrifugal and gravitational force are precompensated for the purpose of achieving model-plant match.

So, total control law U becomes as follows:

$$U = U_a + U_n \tag{5.21}$$

where U_a is an adaptive STC used in Test1 and Test2 for the purpose of *variational control*, and U_n is called *nominal control* with respect to desired position/velocity. Nominal control U_n is created as follows:

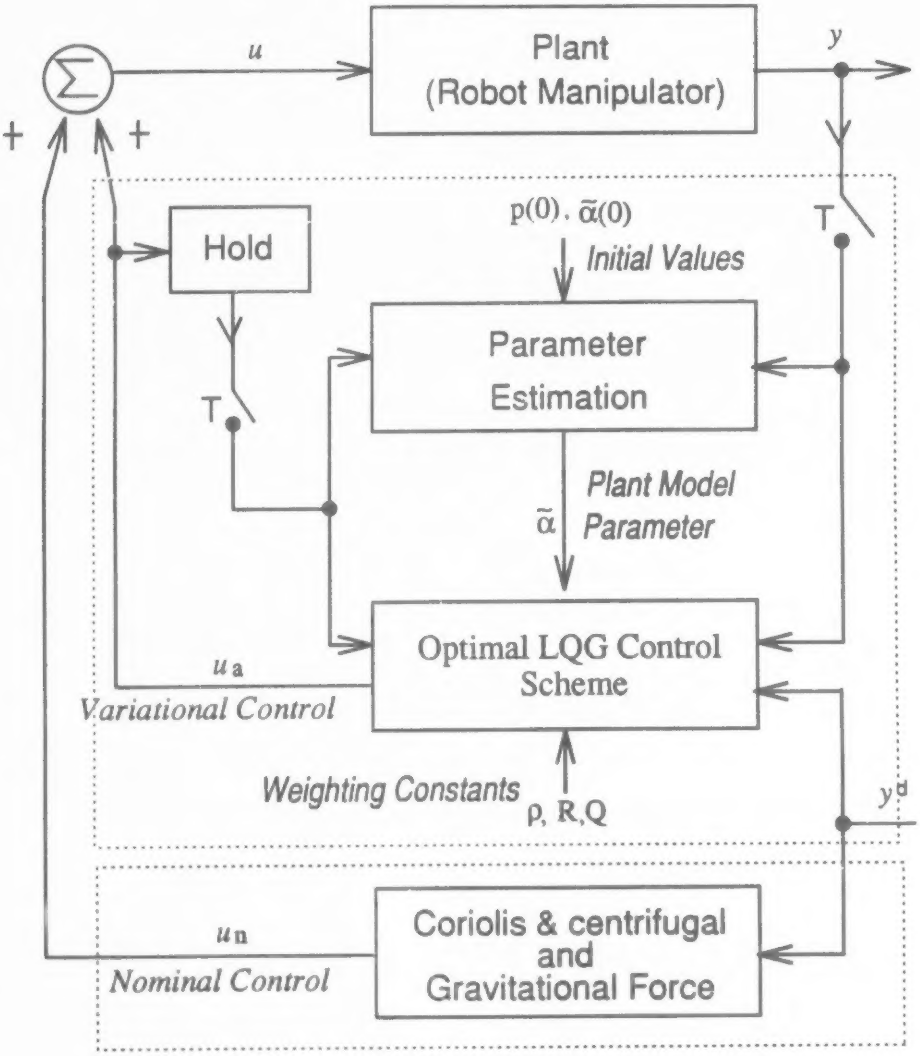
u
Plant
(Robot Manipulator)
y
+
+
Hold
T
p(0), α̃(0)
Initial Values
T
Parameter
Estimation
Plant Model
α̃
Parameter
Optimal LQG Control
Scheme
u_a
Variational Control
Weighting Constants
ρ, R,Q
y^d
Coriolis & centrifugal
and
Gravitational Force
u_n
Nominal Control

Figure 27: Adaptive STC with Weakened Couplings and Non-linearities

$$U_n = C(\Theta^d, \dot{\Theta}^d) + G(\Theta^d) \tag{5.22}$$

where $C(\cdot)$ and $G(\cdot)$ are Coriolis and centrifugal force, and gravitational force in L-E dynamics equation respectively, which are assumed to increase the nonlinearity and coupling effects.

This combination of the nominal control by the computed force method and the variational control by STC results in a robust algorithm with respect to the fast varying dynamics of robot manipulator with a desired high velocity and a payload.

As a conclusion remark, adaptive SIMO STC with both nominal control for the purpose of compensating undesirable dynamics and variational control based on linear model for the unknown dynamics shows good ability to be able to cope with a very fast moving all revolute six-joint manipulator, which has been one of the high-velocity difficulties in controlling the robot manipulator.

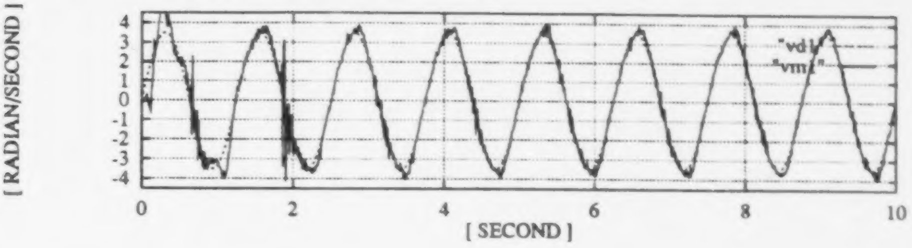

Figure 28: Test3:Velocity Tracking in Joint1 with payload

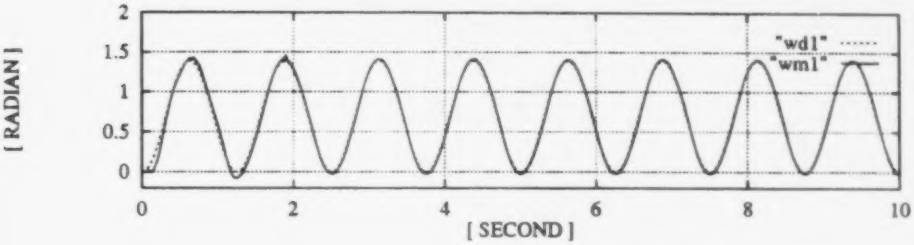

Figure 29: Test3:Position Tracking in Joint1 with payload

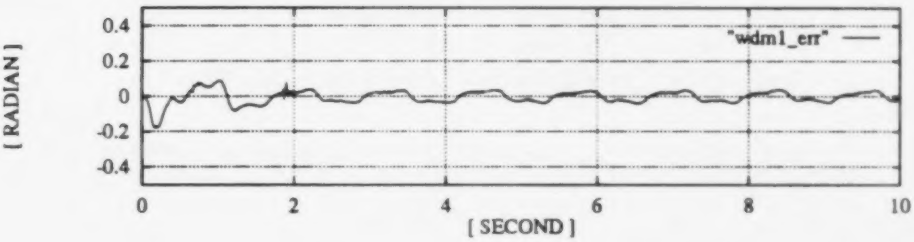

Figure 30: Test3:Position Tracking Error in Joint1 with payload

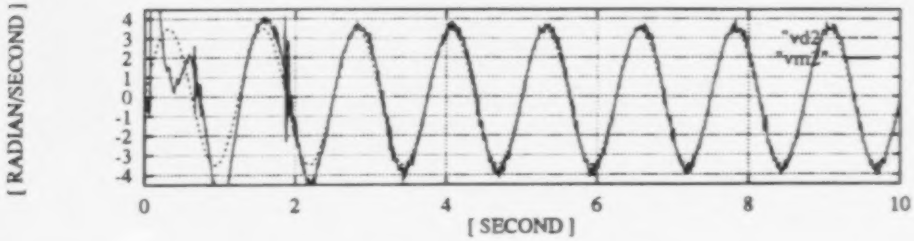

Figure 31: Test3:Velocity Tracking in Joint2 with payload

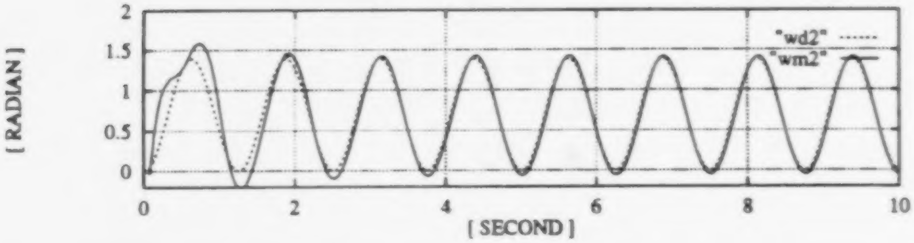

Figure 32: Test3:Position Tracking in Joint2 with payload

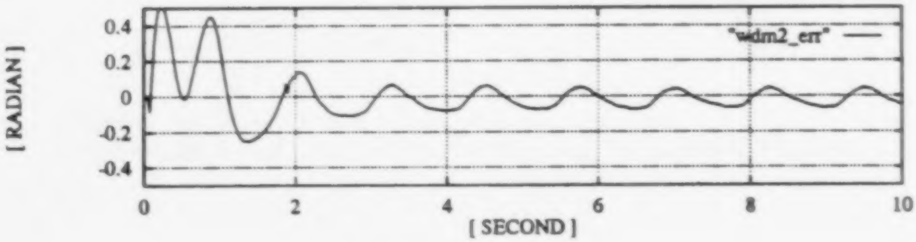

Figure 33: Test3:Position Tracking Error in Joint2 with payload

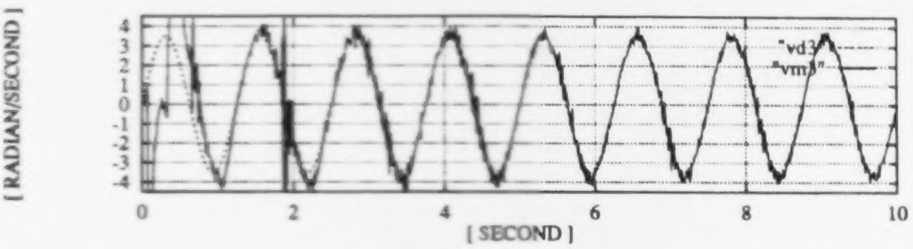

Figure 34: Test3:Velocity Tracking in Joint3 with payload

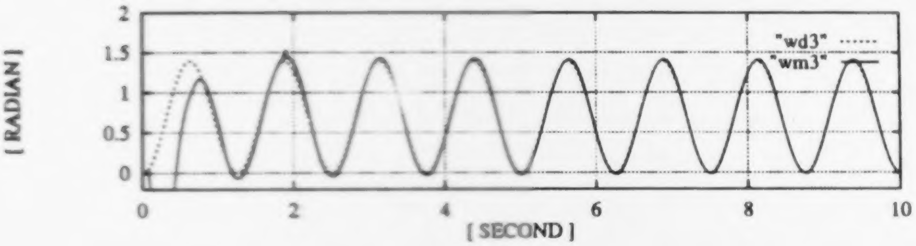

Figure 35: Test3:Position Tracking in Joint3 with payload

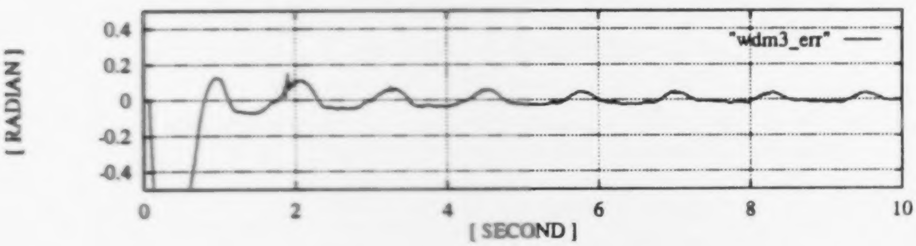

Figure 36: Test3:Position Tracking Error in Joint3 with payload

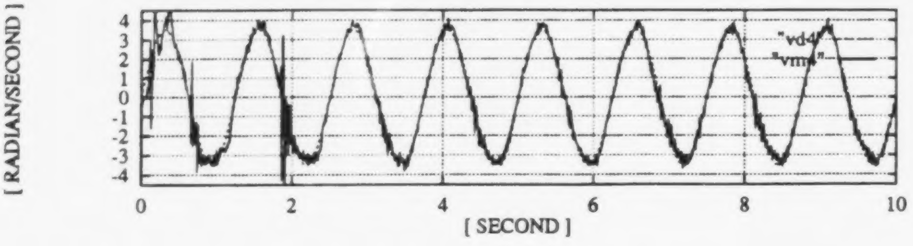

Figure 37: Test3:Velocity Tracking in Joint4 with payload

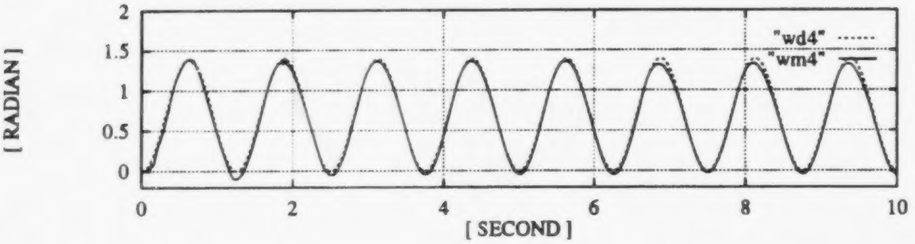

Figure 38: Test3:Position Tracking in Joint4 with payload

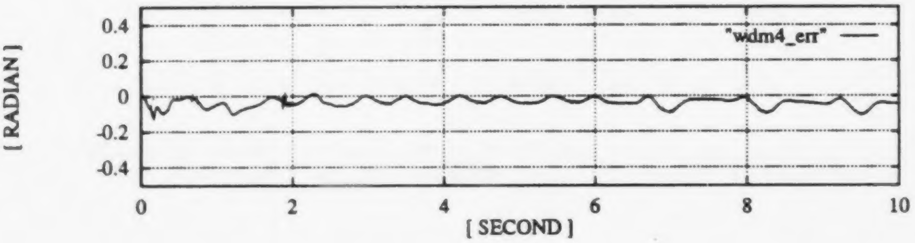

Figure 39: Test3:Position Tracking Error in Joint4 with payload

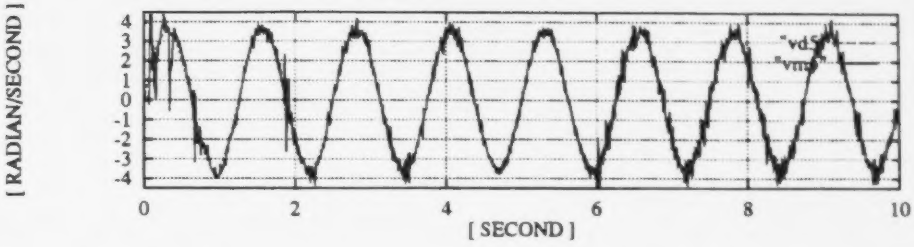

Figure 40: Test3:Velocity Tracking in Joint5 with payload

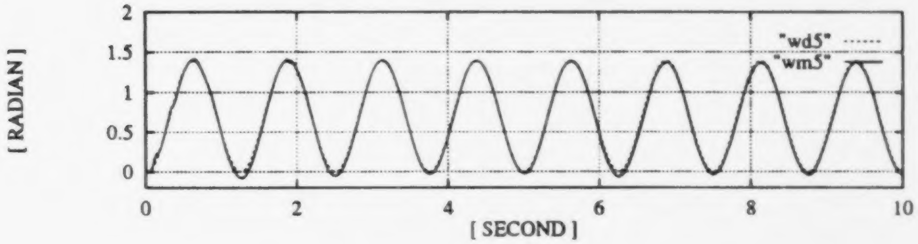

Figure 41: Test3:Position Tracking in Joint5 with payload

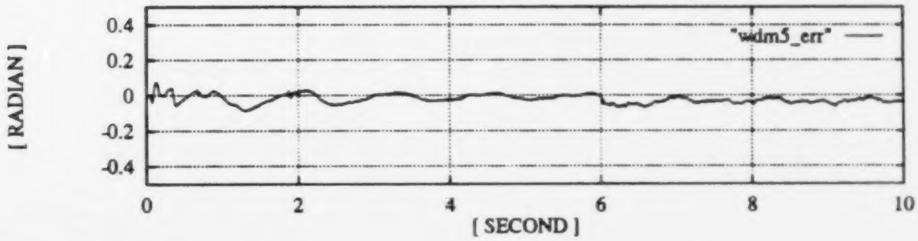

Figure 42: Test3:Position Tracking Error in Joint5 with payload

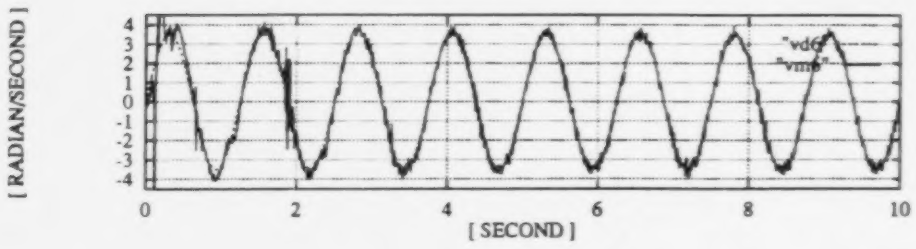

Figure 43: Test3:Velocity Tracking in Joint6 with payload

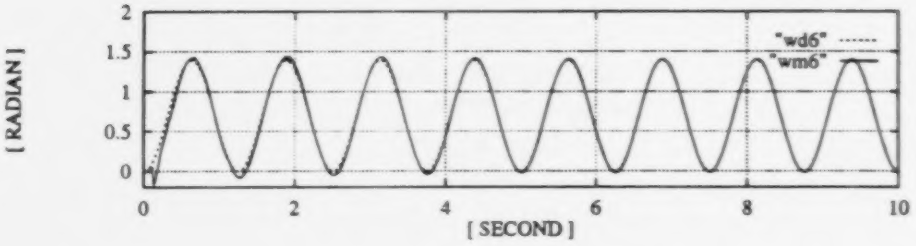

Figure 44: Test3:Position Tracking in Joint6 with payload

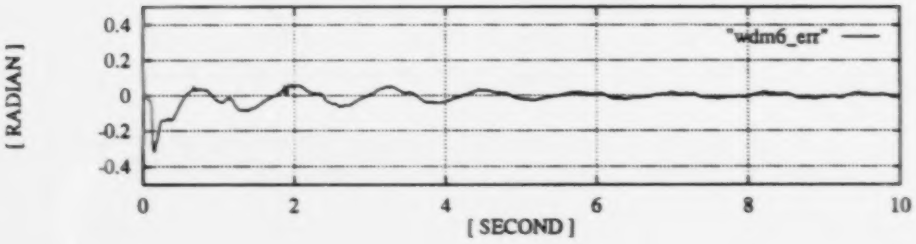

Figure 45: Test3:Position Tracking Error in Joint6 with payload

Chapter 6

Conclusions and Future Work

6.1 Conclusions

An adaptive self-tuning LQG controller suitable for controlling a six-joint robot manipulator working at very high desired (referential) velocity has been proposed and tested on all revolute six joints.

Conventional or even commercial controllers for a robot manipulator have drawbacks in controlling at high velocity because the dynamics of robot is strongly dependent on desired trajectories, and it inherently becomes a non-linear, coupled and time-varying system. In other words, robot dynamics becomes dependent on each joint position, velocity and acceleration variables. For the purpose of overcoming time-varying parameters, non-linearity, couplings between joints, and parameter uncertainty from the unknown payload and dynamics, an adaptive SIMO self-tuning LQG controller was designed and tested. This controller has been extended to one with feedforward (precompensation) scheme for weakening the non-linearity/coupling so that it can cope when the robot manipulator moves at a very high velocity of 3.5 (rad./sec.)

with payload.

Picking up of the payload by the end-effector when moving at a high velocity deteriorates seriously the controller performance. The increased nonlinearity/coupling caused from the dynamics at the desired velocity is suppressed by precompensating the Coriolis/centrifugal and gravitational forces/torques using L-E dynamics equation in the extended adaptive STC. Consequently, the controller described in this thesis performs well for a revolute six-joint robot manipulator which is moving at very high velocity with payload.

6.2 Suggestions for Future Work

The robustness of the adaptive self-tuning controller(STC) is a subject which has generated a great deal of discussion. Since adaptive STC is composed of two parts, parameter estimation and optimal (LQG) controller, if both parts are made to be robust we can make a robust adaptive STC.

In RLS parameter estimation algorithm, which has frequently been used in adaptive STC, forgetting factor(γ) weights the measurements such that a measurement received n samples ago will have a weighting proportional to γ^n. However, constant forgetting factor used in RLS algorithm frequently has some potential implementation difficulties as follows:

- γ must not stay too close to "1" if algorithm is to remain capable of tracking sudden parameter changes.
- On the other hand, when γ is less than "1" *burst phenomenon* - excessive measurement noise, or large spurious variations in estimate owing to the sudden increase in information - may be resulted in.

Methods of adjusting γ automatically in the recursion or using UD factorization technique have been devised for performing better parameter tracking ability [Chen&N 87]. However, those methods should have been tested on the dynamics of robot manipulator instead of on the individual time-varying formula for testing the convergence. Since the parameter estimation algorithm is directly related to the robustness of adaptive STC, the estimation algorithm research is strongly suggested for a robot control by adaptive STC.

According to the proposed controller, joint position tracking error is shown to have increased more at top and bottom of sinusoidal function used for desired velocity profile. That is, a higher desired joint acceleration introduced a larger position tracking error. So, variations of acceleration can be included in cost-function of the optimal LQG controller in order to feedback the effects of joint acceleration into adaptive STC. The systematic method to determine the weighting matrices in cost-function is one of the necessary areas to be tackled in adaptive STC because it influences to the convergence of controller.

In the proposed control method, the non-linear system was controlled by adaptive STC designed based on linear model. Precompensation or feedforward was used in weakening the non-linearity of plant dynamics. Coupling problem was tackled by the MIMO style controller. On the other hand, neural network method may solve obstacles of non-linearity and coupling, which are crucial in robot control, since its structure similar to biological neuron and proper selection of *activation function* for a robot manipulator processes the above obstacles through its neural structure.

Bibliography

[Allen&Y.&T. 90] Allen, P., Yoshimi, B., Timcenko, A., *Real-time Visual Servoing*, Technical Report CUCS-035-90, Department of Computer Science, Columbia University, New York, NY 10027.

[Astrom 80] Astrom, K. J., *Design Principles for Self-Tuning Regulators*, Proc. of International Symposium on Methods and Applications in Adaptive Control, editors, A. V. Balakrishnan and M. Thoma, Bochan, W. Germany, 1980.

[Bejczy 74] Bejczy, A. K., *Robot Arm Dynamics and Control*, Technical Memo 33-669, Jet Propulsion Laboratory, Pasadena, Calif., 1974.

[Bejczy 83] Bejczy, A. K., *Dynamic Analysis for Robot Arm Control*, Proc. 1983 American Control Conf., San Francisco, CA, PP 503-504, 1983.

[Brady 82] Brady, M., *Robot Motion: Planning and Control*, Cambridge, MA, MIT Press, 1982.

[Chen&N 87] Chen, M. J. and Norton, J. P., *Estimation Techniques for Tracking Rapid Parameter Changes*, International J. of Control, Vol. 45, PP. 1387-1398, 1987.

[Craig&H.&S. 86] Craig, J. J., Hsu, P. and Sastry S., *Adaptive Control of Mechanical Manipulators*, IEEE Conference on Robotics and Automation, San Francisco, April, 1986.

[Denavit&H. 55] Denavit, J. and Hartenberg, R. S., *A Kinematic Notation for Lower-Pair Mechanics Based on Matrices*, J. of Applied Mechanics, PP. 215-221, June, 1955.

[Dubowsky&D. 79] Dubowsky, S., and DesForges, D. T., *The Application of Model-Referenced Adaptive Control to Robotic Manipulators*, A.S.M.E. J. Dynam. Syst. Meas. Control 101,193, 1979.

[Honey&J. 92] Honey, W. and Jamshidi, M., *ROBO_SIM: A robotics simulation environment on personal computer*, Int. J. Robotics and Autonomous Systems, Vol.9, No.4, 1992.

[Huston&K. 82] Huston, R. L. and Kelly, F. A., *The Development of Equations of Equations of Motion of Single-Arm Robots*, IEEE Trans., Systems, Man, and Cybernetics, Vol. SMC-12, No. 3, PP. 259-266, 1982.

[Khorrami&O. 88] Khorrami, F. and Ozguner, U., *Decentralized Control of Robot Manipulators Via State and Proportional-Integral Feedback*, Proceedings of the 1988 IEEE International Conference on Robotics and automation, Philadelphia, PA, USA, Apr.24-29, 1988.

[Koivo&G. 83] Koivo, A. J. and Guo, T. H., *Adaptive Linear Controller for Robotic Manipulators*, IEEE Transactions on Automatic Control, Vol. AC-28, No.2, PP. 162-171, Feb., 1983.

[Koivo&H. 91] Koivo, A. J. and Houshangi, N., *Real-Time Vision Feedback For Servoing Robotic Manipulator With Self-Tuning Controller*, IEEE

Transactions on Systems, Man and Cybernetics, Vol.21 No.21, Jan./Feb., 1991.

[Lee 82] Lee, C. S., *Robotic Arm Kinematics Dynamics and control*, Computer, IEEE, PP. 62-80, December, 1982.

[Lee&C. 84] Lee, C. S. G. and Chung, M. J., *An Adaptive Control Strategy for Mechanical Manipulators*, IEEE Trans. Automatic Control, Vol. AC-29, No.9, PP. 837-840, 1984.

[Lee&C. 85] Lee, C. S. G. and Chung, M. J., *An Adaptive Perturbation Control with Feedforward Compensation for Robot Manipulators*, Simulation, Vol.44 No.3, PP. 127-136, 1985.

[Lee&G.&F. 87] Lee, C. S. G., Gonzzalez, R. C. and Fu, K. S., *Robotics: Control, Sensing, Vision, and Intelligence*, Singapore, McGraw-Hill Book Co., 1987.

[Lewis 74] Lewis, R. A., *Autonomous Manipulation on a Robot: Summary of Manipulator Software Functions*, Jet Propulsion Lab., Pasadena, CA, Tech. Memo. 33-69, Mar. 1974.

[Ljung 83] Ljung, L. and Soderstrom, T., *Theory and Practice of Recursive Identification*, MIT Press, London, 1983.

[Maple 91] Symbolic Computation Group, *Maple V*, The University of Waterloo, Ont., Canada.

[Murray&N. 84] Murray, J. J. and Neuman, C. P., *ARM: An Algebraic Robot Dynamic Modeling Program*, Proc. of 1984 IEEE International Conference on Robotics, Atlanta, PP. 103-114, 1984.

[Neuman&M. 85] Neuman, C. P. and Murray, J. J., *Computational robot dynamics: Foundations and applications*, J. Robotic Syst., Vol. 2, PP. 425-452, Winter 1985.

[Neuman&M. 87] Neuman, C. P. and Murray, J. J., *The Complete Dynamic Model and Customized Algorithms of the Puma Robot*, IEEE Trans. on Systems, Man, and Cybernetics, Vol. SMC-17, No. 4, PP. 635-644, July/August 1987.

[Paul 81] Paul, R. P., *Robot Manipulators: Mathematics, Programming, and Control*, Cambridge, MA: MIT Press, 1981.

[Paul&R.&Z. 83] Paul, R. P., Rong, M. and Zhang, H., *The Dynamics of the PUMA manipulator*, Proc. 1983 American Control Conf., San Francisco, CA., PP. 491-496, June, 1983.

[Press&F.&T.&V. 88] Press, W. H., Flannery, B. P., Teukolsky, S. A. and Vetterling, W. T., *Numerical Recipes in C : The Art of Scientific Computing* , Cambridge University Press 1988.

[Souissi&K. 87] Souissi, R. and Koivo, A. J., *Design of Adaptive Self-tuning Controller for High Speed Manipulators*, Proceedings of the 1987 IEEE International Conference on Systems, Man and Cybernetics, Vol.3, Alexandria, Virginia, Oct. 20-23, 1987.

[Takegaki&A. 80] Takegaki, M., and Arimoto, S., *An Adaptive Trajectory Control of Manipulators*, Int. J. 34,219, 1981.

[Tourassis&N. 85] Tourassis, T. D. and Neuman, C. P., *The Inertial Characteristics of Dynamic Robot Models*, Mechanism and machine Theory, Vol. 20, No. 1985, PP. 41-52.

[Uicker 65] Uicker, J. J., *On the Dynamic Analysis of Spatial Linkages Using 4x4 Matrices*, Ph.D. Dissertation, Northwestern University, Evanston, 1965.

[Unimate 85] A Westinghouse Company, *500 Series Equipment Manual for VAL* II *and VAL PLUS Operating Systems*, 1985.

[Walker&O. 82] Walker, M. W. and Orin, D. E., *Efficient Dynamic Computer Simulation of Robotic Mechanisms*, Trans., ASME, J. Systems, Measurement and Control, Vol. 104, PP. 205-211, 1982.

Appendix A

Customized L-E Forward Dynamics Equation for six-joint PUMA 600

The following equations are written in C program language.

"puma_f_dynamics.c"

```
/*********************************************************************
 *                                                                   *
 *  Function, *PUMA_F_Dynamics() calculates (6x1)angular-acceleration *
 *  vector from given (6x1)angular-postion, (6x1)angular-velocity and *
 *  (6x1)generalized-force vector for six-joint PUMA 600. That is,    *
 *  FORWARD DYNAMICS EQUATION based on Lagrange-Euler dynamics model. *
 *                                                                   *
 *********************************************************************/
#include<stdio.h>
double *PUMA_F_Dynamics(row1,col1,tau,row2,col2,theta,row3,col3,omega)
 int row1, /* Row of vector, generalized-force(tau) == 6 */
     col1, /* Column of vector tau == 1 */
     row2, /* Row of vector, angular-position(theta) == 6 */
     col2, /* Column of theta == 1 */
     row3, /* Row of vector, angular-velocity(omega) == 6 */
```

```
    col3; /* Column of omega == 1 */
double
    *tau,   /* Generalized force(input to manipulator) vector */
    *theta, /* Joint angular-position vector */
    *omega; /* Joint angular-velocity vector */

{
  double     *calloc(),*D_Matrix,*C_Vector,*G_Vector,
             *Acc_Angle_Vector, /* Vector of angular-acceleration */
             *Make_D_Matrix(),*Make_C_Vector(),*Make_G_Vector(),
             *Matrix_Inv(),*Matrix2_Mul(),*Matrix3_Sub(),*D_inverse,
             *temp1;
  if(row1!=row2 || row2!=row3 || col1!=col2 || col2!=col3 || row1!=6
     || col1!=1){
     printf("Dimension errors of arguments in PUMA_F_Dynamics().\n");
     exit();
  }
 D_Matrix=Make_D_Matrix(row2,col2,theta);
 C_Vector=Make_C_Vector(row2,col2,theta,row3,col3,omega);
 G_Vector=Make_G_Vector(row2,col2,theta);
 D_inverse=Matrix_Inv(row2,row2,D_Matrix);
 temp1=Matrix3_Sub(row1,col1,tau,row2,col2,C_Vector,row2,col2,G_Vector);
 Acc_Angle_Vector=Matrix2_Mul(row2,row2,D_inverse,row1,col1,temp1);
 free(D_Matrix);free(C_Vector);free(G_Vector);
 free(D_inverse);free(temp1);
 return(Acc_Angle_Vector);
}
```

"make_d_matrix.c"

```
#include<stdio.h>
#include <math.h>
#include "PUMA_SPEC_DEF.h"

double *Make_D_Matrix(row,one,pos_angle)
int row,one;
double *pos_angle;
{
  double *d_matrix,p1,p2,p3,p4,p5,p6;

  if(row!=6 || one!=1){
     printf("Dimension error in Make_D_Matrix().\n");
     exit();
  }
/* calloc() returns pointer of specified size whose contents */
/* are zero.                                                 */
  d_matrix = (double *) calloc(row*row, sizeof(double));
  if(d_matrix == NULL){
     printf("calloc() returned NULL in Make_D_Matrix().\n");
     exit();
  }

  p1= *(pos_angle);   p2= *(pos_angle+1); p3= *(pos_angle+2);
  p4= *(pos_angle+3); p5= *(pos_angle+4); p6= *(pos_angle+5);

/*** D[1,1] ***/
*d_matrix=2*a3*d4*m6*c(p2+p3)*s(p2+p3)+2*a2*d4*m6*c(p2)*s(p2+p3)
  +sq(d4)*m6*s2(p2+p3) +2*a2*a3*m6*c(p2)*c(p2+p3)+sq(a2)*m6
  *c2(p2)+sq(d3)*m6+sq(a3)*m6*c2(p2+p3)+J6zz*s2(p4)*s2(p5)+J6zz
  *c2(p5)*s2(p2+p3)+2*J6zz*c(p2+p3)*c(p4)*c(p5)*s(p2+p3)*s(p5)+J6zz
  *c2(p2+p3)*c2(p4)*s2(p5)+2*a3*m6*R6z*c(p2+p3)*c(p5)*s(p2+p3)
  +2*a3*m6*R6z*c2(p2+p3)*c(p4)*s(p5)+2*a2*m6*R6z*c(p2)*c(p5)
  *s(p2+p3)+2*a2*m6*R6z*c(p2)*c(p2+p3)*c(p4)*s(p5)+2*d4*m6*R6z
  *c(p5)*s2(p2+p3)+2*d4*m6*R6z*c(p2+p3)*c(p4)*s(p2+p3)*s(p5)+2
  *d3*m6*R6z*s(p4)*s(p5)+J6yy*c2(p4)*c2(p6)+J6yy*c2(p5)*s2(p4)
  *s2(p6)+J6yy*s2(p2+p3)*s2(p5)*s2(p6)-2*J6yy*c(p2+p3)*c(p6)
  *s(p2+p3)*s(p4)*s(p5)*s(p6)+J6yy*c2(p2+p3)*c2(p6)*s2(p4)
  -2*J6yy*c(p2+p3)*c(p4)*c(p5)*s(p2+p3)*s(p5)*s2(p6)-2*J6yy
  *c(p4)*c(p5)*c(p6)*s2(p2+p3)*s(p4)*s(p6)+J6yy*c2(p2+p3)*c2(p4)
  *c2(p5)*s2(p6)+J6xx*c2(p4)*s2(p6)+J6xx*c2(p2+p3)*s2(p4)*s2(p6)
  +2*J6xx*c(p2+p3)*c(p6)*s(p2+p3)*s(p4)*s(p5)*s(p6)+J6xx*c2(p6)
```

```
*s2(p2+p3)*s2(p5)+J6xx*c2(p5)*c2(p6)*s2(p4)+2*J6xx*c(p4)*c(p5)
*c(p6)*s2(p2+p3)*s(p4)*s(p6)-2*J6xx*c(p2+p3)*c(p4)*c(p5)*c2(p6)
*s(p2+p3)*s(p5)+J6xx*c2(p2+p3)*c2(p4)*c2(p5)*c2(p6)+2*a3*d4*m4
*c(p2+p3)*s(p2+p3)+2*a2*d4*m4*c(p2)*s(p2+p3)+sq(d4)*m4
*s2(p2+p3)+2*a2*a3*m4*c(p2)*c(p2+p3)+sq(a2)*m4*c2(p2)+sq(d3)
*m4+sq(a3)*m4*c2(p2+p3)+J4zz*c2(p4)+J4zz*c2(p2+p3)*s2(p4)+J4yy
*s2(p2+p3)-2*a3*m4*R4y*c(p2+p3)*s(p2+p3)-2*a2*m4*R4y*c(p2)
*s(p2+p3)-2*d4*m4*R4y*s2(p2+p3)+J4xx*s2(p4)+J4xx*c2(p2+p3)
*c2(p4)+sq(a2)*m2*c2(p2)+J2zz+J2yy*s2(p2)+2*a2*m2*R2x*c2(p2)
+J2xx*c2(p2)+J1xx+J1zz+J3xx*c2(p2+p3)+J3yy+2*a3*m3*R3z*c(p2+p3)
*s(p2+p3)+2*a2*m3*R3z*c(p2)*s(p2+p3)+J3zz*s2(p2+p3)+sq(a2)*m3
*c2(p2)+sq(d3)*m3+sq(a3)*m3*c2(p2+p3)+2*a2*a3*m3*c(p2)*c(p2+p3)
+J5xx*c2(p2+p3)*c2(p4)*c2(p5)-2*J5xx*c(p2+p3)*c(p4)*c(p5)
*s(p2+p3)*s(p5)+J5xx*c2(p5)*s2(p4)+J5xx*s2(p2+p3)*s2(p5)+J5yy
*c2(p2+p3)*s2(p4)+J5yy*c2(p4)+2*d3*m5*R5z*s(p4)*s(p5)+2*d4*m5
*R5z*c(p2+p3)*c(p4)*s(p2+p3)*s(p5)+2*d4*m5*R5z*c(p5)*s2(p2+p3)
+2*a2*m5*R5z*c(p2)*c(p2+p3)*c(p4)*s(p5)+2*a2*m5*R5z*c(p2)*c(p5)
*s(p2+p3)+2*a3*m5*R5z*c2(p2+p3)*c(p4)*s(p5)+2*a3*m5*R5z*c(p2+p3)
*c(p5)*s(p2+p3)+J5zz*c2(p2+p3)*c2(p4)*s2(p5)+2*J5zz*c(p2+p3)*c(p4)
*c(p5)*s(p2+p3)*s(p5)+J5zz*c2(p5)*s2(p2+p3)+J5zz*s2(p4)*s2(p5)
+sq(a3)*m5*c2(p2+p3)+sq(d3)*m5+sq(a2)*m5*c2(p2)+2*a2*a3*m5*c(p2)
*c(p2+p3)+sq(d4)*m5*s2(p2+p3)+2*a2*d4*m5*c(p2)*s(p2+p3)+2*a3*d4
*m5*c(p2+p3)*s(p2+p3);

/*** D[1,2] ***/
*(d_matrix+1)=-d3*m6*R6z*c(p2+p3)*c(p5)+d3*m6*R6z*c(p4)*s(p2+p3)
  *s(p5)+a3*d3*m6*s(p2+p3)+a2*d3*m6*s(p2)-d3*d4*m6*c(p2+p3)-J6zz
  *c(p2+p3)*c(p5)*s(p4)*s(p5)+J6zz*c(p4)*s(p2+p3)*s(p4)*s2(p5)+a3
  *m6*R6z*s(p2+p3)*s(p4)*s(p5)+a2*m6*R6z*s(p2)*s(p4)*s(p5)-d4*m6
  *R6z*c(p2+p3)*s(p4)*s(5)-J6yy*c(p2+p3)*c(p4)*c(p6)*s(p5)*s(p6)
  -J6yy*c(p4)*c2(p6)*s(p2+p3)*s(p4)+J6yy*c(p2+p3)*c(p5)*s(p4)*s(p5)
  *s2(p6)+J6yy*c(p5)*c(p6)*s(p2+p3)*s(p6)-2*J6yy*c2(p4)*c(p5)*c(p6)
  *s(p2+p3)*s(p6)+J6yy*c(p4)*c2(p5)*s(p2+p3)*s(p4)*s2(p6)-J6xx*c(p4)
  *s(p2+p3)*s(p4)*s2(p6)+J6xx*c(p2+p3)*c(p4)*c(p6)*s(p5)*s(p6)+J6xx
  *c(p2+p3)*c(p5)*c2(p6)*s(p4)*s(p5)-J6xx*c(p5)*c(p6)*s(p2+p3)*s(p6)
  +2*J6xx*c2(p4)*c(p5)*c(p6)*s(p2+p3)*s(p6)+J6xx*c(p4)*c2(p5)*c2(p6)
  *s(p2+p3)*s(p4)+d3*m4*R4y*c(p2+p3)+a3*d3*m4*s(p2+p3)+a2*d3*m4
  *s(p2)-d3*d4*m4*c(p2+p3)-J4zz*c(p4)*s(p2+p3)*s(p4)+J4xx*c(p4)
  *s(p2+p3)*s(p4)+a2*m2*R2z*s(p2)+a3*d3*m3*s(p2+p3)+a2*d3*m3*s(p2)
  -d3*m3*R3z*c(p2+p3)+J5xx*c(p4)*c2(p5)*s(p2+p3)*s(p4)+J5xx*c(p2+p3)
  *c(p5)*s(p4)*s(p5)-J5yy*c(p4)*s(p2+p3)*s(p4)-d4*m5*R5z*c(p2+p3)
  *s(p4)*s(p5)+a2*m5*R5z*s(p2)*s(p4)*s(p5)+a3*m5*R5z*s(p2+p3)*s(p4)
  *s(p5)+J5zz*c(p4)*s(p2+p3)*s(p4)*s2(p5)-J5zz*c(p2+p3)*c(p5)*s(p4)
  *s(p5)-d3*d4*m5*c(p2+p3)+a2*d3*m5*s(p2)+a3*d3*m5*s(p2+p3)+d3*m5
  *R5z*c(p4)*s(p2+p3)*s(p5)-d3*m5*R5z*c(p2+p3)*c(p5);

/*** D[2,1] ***/
*(d_matrix+row)= *(d_matrix+1);

/*** D[2,2] ***/
*(d_matrix+row+1)=2*a2*d4*m6*s(p3)+sq(d4)*m6+2*a2*a3*m6*c(p3)
  +sq(a2)*m6+sq(a3)*m6+J6zz*c2(p5)+J6zz*c2(p4)*s2(p5)+2*a3*m6
  *R6z*c(p4)*s(p5)+2*a2*m6*R6z*c(p5)*s(p3)+2*a2*m6*R6z*c(p3)
  *c(p4)*s(p5)+2*d4*m6*R6z*c(p5)+J6yy*s2(p5)*s2(p6)+J6yy*c2(p6)
  *s2(p4)+2*J6yy*c(p4)*c(p5)*c(p6)*s(p4)*s(p6)+J6yy*c2(p4)*c2(p5)
  *s2(p6)+J6xx*s2(p4)*s2(p6)+J6xx*c2(p6)*s2(p5)-2*J6xx*c(p4)
  *c(p5)*c(p6)*s(p4)*s(p6)+J6xx*c2(p4)*c2(p5)*c2(p6)+2*a2*d4
  *m4*s(p3)+sq(d4)*m4+2*a2*a3*m4*c(p3)+sq(a2)*m4+sq(a3)*m4+J4zz
  *s2(p4)+J4yy-2*a2*m4*R4y*s(p3)-2*d4*m4*R4y+J4xx*c2(p4)+sq(a2)
  *m2+J2yy+2*a2*m2*R2x+J2xx+J3xx+2*a2*m3*R3z*s(p3)+J3zz+sq(a2)
  *m3+sq(a3)*m3+2*a2*a3*m3*c(p3)+J5xx*c2(p4)*c2(p5)+J5xx*s2(p5)
  +J5yy*s2(p4)+2*d4*m5*R5z*c(p5)+2*a2*m5*R5z*c(p3)*c(p4)*s(p5)
  +2*a2*m5*R5z*c(p5)*s(p3)+2*a3*m5*R5z*c(p4)*s(p5)+J5zz*c2(p4)
  *s2(p5)+J5zz*c2(p5)+sq(a3)*m5+sq(a2)*m5+2*a2*a3*m5*c(p3)+sq(d4)
  *m5+2*a2*d4*m5*s(p3);

/*** D[1,3] ***/
*(d_matrix+2)=-d3*m6*R6z*c(p2+p3)*c(p5)+d3*m6*R6z*c(p4)*s(p2+p3)
  *s(p5)+a3*d3*m6*s(p2+p3)-d3*d4*m6*c(p2+p3)-J6zz*c(p2+p3)*c(p5)
  *s(p4)*s(p5)+J6zz*c(p4)*s(p2+p3)*s(p4)*s2(p5)+a3*m6*R6z*s(p2+p3)
  *s(p4)*s(p5)-d4*m6*R6z*c(p2+p3)*s(p4)*s(p5)-J6yy*c(p2+p3)*c(p4)
  *c(p6)*s(p5)*s(p6)-J6yy*c(p4)*c2(p6)*s(p2+p3)*s(p4)+J6yy*c(p2+p3)
  *c(p5)*s(p4)*s(p5)*s2(p6)+J6yy*c(p5)*c(p6)*s(p2+p3)*s(p6)-2*J6yy
  *c2(p4)*c(p5)*c(p6)*s(p2+p3)*s(p6)+J6yy*c(p4)*c2(p5)*s(p2+p3)
  *s(p4)*s2(p6)-J6xx*c(p4)*s(p2+p3)*s(p4)*s2(p6)+J6xx*c(p2+p3)
  *c(p4)*c(p6)*s(p5)*s(p6)+J6xx*c(p2+p3)*c(p5)*c2(p6)*s(p4)*s(p5)
  -J6xx*c(p5)*c(p6)*s(p2+p3)*s(p6)+2*J6xx*c2(p4)*c(p5)*c(p6)
  *s(p2+p3)*s(p6)+J6xx*c(p4)*c2(p5)*c2(p6)*s(p2+p3)*s(p4)+d3*m4
  *R4y*c(p2+p3)+a3*d3*m4*s(p2+p3)-d3*d4*m4*c(p2+p3)-J4zz*c(p4)
  *s(p2+p3)*s(p4)+J4xx*c(p4)*s(p2+p3)*s(p4)+a3*d3*m3*s(p2+p3)
  -d3*m3*R3z*c(p2+p3)+J5xx*c(p4)*c2(p5)*s(p2+p3)*s(p4)+J5xx
  *c(p2+p3)*c(p5)*s(p4)*s(p5)-J5yy*c(p4)*s(p2+p3)*s(p4)-d4*m5*R5z
  *c(p2+p3)*s(p4)*s(p5)+a3*m5*R5z*s(p2+p3)*s(p4)*s(p5)+J5zz*c(p4)
  *s(p2+p3)*s(p4)*s2(p5)-J5zz*c(p2+p3)*c(p5)*s(p4)*s(p5)-d3*d4*m5
  *c(p2+p3)+a3*d3*m5*s(p2+p3)+d3*m5*R5z*c(p4)*s(p2+p3)*s(p5)-d3
  *m5*R5z*c(p2+p3)*c(p5);

/*** D[3,1] ***/
*(d_matrix+(row*2))= *(d_matrix+2);

/*** D[2,3] ***/
*(d_matrix+row+2)=sq(d4)*m6+a2*m6*R6z*c(p5)*s(p3)+a2*m6*R6z*c(p3)
  *c(p4)*s(p5)+a2*a3*m6*c(p3)+a2*d4*m6*s(p3)+sq(a3)*m6+J6zz*c2(p5)
  +J6zz*c2(p4)*s2(p5)+2*a3*m6*R6z*c(p4)*s(p5)+2*d4*m6*R6z*c(p5)
  +J6yy*s2(p5)*s2(p6)+J6yy*c2(p6)*s2(p4)+2*J6yy*c(p4)*c(p5)*c(p6)
  *s(p4)*s(p6)+J6yy*c2(p4)*c2(p5)*s2(p6)+J6xx*s2(p4)*s2(p6)+J6xx
  *c2(p6)*s2(p5)-2*J6xx*c(p4)*c(p5)*c(p6)*s(p4)*s(p6)+J6xx*c2(p4)
  *c2(p5)*c2(p6)+sq(d4)*m4-a2*m4*R4y*s(p3)+a2*a3*m4*c(p3)+a2*d4
  *m4*s(p3)+sq(a3)*m4+J4zz*s2(p4)+J4yy-2*d4*m4*R4y+J4xx*c2(p4)
  +J3xx+J3zz+a2*a3*m3*c(p3)+a2*m3*R3z*s(p3)+sq(a3)*m3+J5xx*c2(p4)
  *c2(p5)+J5xx*s2(p5)+J5yy*s2(p4)+2*d4*m5*R5z*c(p5)+2*a3*m5*R5z
  *c(p4)*s(p5)+J5zz*c2(p4)*s2(p5)+J5zz*c2(p5)+sq(a3)*m5+a2*d4*m5
  *s(p3)+a2*a3*m5*c(p3)+a2*m5*R5z*c(p3)*c(p4)*s(p5)+a2*m5*R5z*c(p5)
  *s(p3)+sq(d4)*m5;

/*** D[3,2] ***/
*(d_matrix+(row*2)+1)= *(d_matrix+row+2);

/*** D[3,3] ***/
*(d_matrix+(row*2)+2)=sq(d4)*m6+sq(a3)*m6+J6zz*c2(p5)+J6zz*c2(p4)
  *s2(p5)+2*a3*m6*R6z*c(p4)*s(p5)+2*d4*m6*R6z*c(p5)+J6yy*s2(p5)
  *s2(p6)+J6yy*c2(p6)*s2(p4)+2*J6yy*c(p4)*c(p5)*c(p6)*s(p4)*s(p6)
  +J6yy*c2(p4)*c2(p5)*s2(p6)+J6xx*s2(p4)*s2(p6)+J6xx*c2(p6)*s2(p5)
  -2*J6xx*c(p4)*c(p5)*c(p6)*s(p4)*s(p6)+J6xx*c2(p4)*c2(p5)*c2(p6)
  +sq(d4)*m4+sq(a3)*m4+J4zz*s2(p4)+J4yy-2*d4*m4*R4y+J4xx*c2(p4)+J3xx
  +J3zz+sq(a3)*m3+J5xx*c2(p4)*c2(p5)+J5xx*s2(p5)+J5yy*s2(p4)+2*d4*m5
  *R5z*c(p5)+2*a3*m5*R5z*c(p4)*s(p5)+J5zz*c2(p4)*s2(p5)+J5zz*c2(p5)
  +sq(a3)*m5+sq(d4)*m5;

/*** D[1,4] ***/
*(d_matrix+3)=d4*m6*R6z*c(p4)*s(p2+p3)*s(p5)+a2*m6*R6z*c(p2)*c(p4)
  *s(p5)+d3*m6*R6z*c(p2+p3)*s(p4)*s(p5)+a3*m6*R6z*c(p2+p3)*c(p4)
  *s(p5)+J6zz*c(p4)*c(p5)*s(p2+p3)*s(p5)+J6zz*c(p2+p3)*s2(p5)-J6yy
  *c(p6)*s(p2+p3)*s(p4)*s(p5)*s(p6)+J6yy*c(p2+p3)*c2(p6)-J6yy*c(p4)
  *c(p5)*s(p2+p3)*s(p5)*s2(p6)+J6yy*c(p2+p3)*c2(p5)*s2(p6)+J6xx
  *c(p2+p3)*s2(p6)+J6xx*c(p6)*s(p2+p3)*s(p4)*s(p5)*s(p6)-J6xx*c(p4)
```

```
   *c(p5)*c2(p6)*s(p2+p3)*s(p5)+J6xx*c(p2+p3)*c2(p5)*c2(p6)+J4zz
   *c(p2+p3)+J4xx*c(p2+p3)+J5xx*c(p2+p3)*c2(p5)-J5xx*c(p4)*c(p5)
   *s(p2+p3)*s(p5)+J5yy*c(p2+p3)+J5zz*c(p2+p3)*s2(p5)+J5zz*c(p4)
   *c(p5)*s(p2+p3)*s(p5)+a3*m5*R5z*c(p2+p3)*c(p4)*s(p5)+d3*m5*R5z
   *c(p2+p3)*s(p4)*s(p5)+a2*m5*R5z*c(p2)*c(p4)*s(p5)+d4*m5*R5z*c(p4)
   *s(p2+p3)*s(p5);

/*** D[4,1] ***/
*(d_matrix+(row*3))= *(d_matrix+3);

/*** D[2,4] ***/
*(d_matrix+row+3)=-d4*m6*R6z*s(p4)*s(p5)-a2*m6*R6z*s(p3)*s(p4)*s(p5)
   -J6zz*c(p5)*s(p4)*s(p5)-J6yy*c(p4)*c(p6)*s(p5)*s(p6)+J6yy*c(p5)
   *s(p4)*s(p5)*s2(p6)+J6xx*c(p4)*c(p6)*s(p5)*s(p6)+J6xx*c(p5)*c2(p6)
   *s(p4)*s(p5)+J5xx*c(p5)*s(p4)*s(p5)-J5zz*c(p5)*s(p4)*s(p5)-a2*m5
   *R5z*s(p3)*s(p4)*s(p5)-d4*m5*R5z*s(p4)*s(p5);

/*** D[[4,2] ***/
*(d_matrix+(row*3)+1)= *(d_matrix+row+3);

/*** D[3,4] ***/
*(d_matrix+(row*2)+3)=-d4*m6*R6z*s(p4)*s(p5)-J6zz*c(p5)*s(p4)*s(p5)
   -J6yy*c(p4)*c(p6)*s(p5)*s(p6)+J6yy*c(p5)*s(p4)*s(p5)*s2(p6)+J6xx
   *c(p4)*c(p6)*s(p5)*s(p6)+J6xx*c(p5)*c2(p6)*s(p4)*s(p5)+J5xx*c(p5)
   *s(p4)*s(p5)-J5zz*c(p5)*s(p4)*s(p5)-d4*m5*R5z*s(p4)*s(p5);

/*** D[4,3] ***/
*(d_matrix+(row*3)+2)= *(d_matrix+(row*2)+3);

/*** D[4,4] ***/
*(d_matrix+(row*3)+3)=J6zz*s2(p5)+J6yy*c2(p6)+J6yy*c2(p5)*s2(p6)
   +J6xx*s2(p6)+J6xx*c2(p5)*c2(p6)+J4zz+J4xx+J5xx*c2(p5)+J5yy+J5zz
   *s2(p5);

/*** D[1,5] ***/
*(d_matrix+4)=d4*m6*R6z*c(p5)*s(p2+p3)*s(p4)+a2*m6*R6z*c(p2)*c(p5)
   *s(p4)+d3*m6*R6z*s(p2+p3)*s(p5)-d3*m6*R6z*c(p2+p3)*c(p4)*c(p5)+a3
   *m6*R6z*c(p2+p3)*c(p5)*s(p4)+J6zz*s(p2+p3)*s(p4)-J6yy*c(p4)*c(p5)
   *c(p6)*s(p2+p3)*s(p6)+J6yy*s(p2+p3)*s(p4)*s2(p6)-J6yy*c(p2+p3)*c(p6)
   *s(p5)*s(p6)+J6xx*c(p4)*c(p5)*c(p6)*s(p2+p3)*s(p6)+J6xx*c(p2+p3)
   *c(p6)*s(p5)*s(p6)+J6xx*c2(p6)*s(p2+p3)*s(p4)+J5xx*s(p2+p3)*s(p4)
   +J5zz*s(p2+p3)*s(p4)+a3*m5*R5z*c(p2+p3)*c(p5)*s(p4)-d3*m5*R5z
```

```
  *c(p2+p3)*c(p4)*c(p5)+d3*m5*R5z*s(p2+p3)*s(p5)+a2*m5*R5z*c(p2)
  *c(p5)*s(p4)+d4*m5*R5z*c(p5)*s(p2+p3)*s(p4);

/*** D[5,1] ***/
*(d_matrix+(row*4))= *(d_matrix+4);

/*** D[2,5] ***/
*(d_matrix+row+4)=d4*m6*R6z*c(p4)*c(p5)+a2*m6*R6z*c(p3)*s(p5)+a2*m6
  *R6z*c(p4)*c(p5)*s(p3)+a3*m6*R6z*s(p5)+J6zz*c(p4)+J6yy*c(p5)*c(p6)
  *s(p4)*s(p6)+J6yy*c(p4)*s2(p6)-J6xx*c(p5)*c(p6)*s(p4)*s(p6)+J6xx
  *c(p4)*c2(p6)+J5xx*c(p4)+J5zz*c(p4)+a3*m5*R5z*s(p5)+a2*m5*R5z*c(p4)
  *c(p5)*s(p3)+a2*m5*R5z*c(p3)*s(p5)+d4*m5*R5z*c(p4)*c(p5);

/*** D[5,2] ***/
*(d_matrix+(row*4)+1)= *(d_matrix+row+4);

/*** D[3,5] ***/
*(d_matrix+(row*2)+4)=d4*m6*R6z*c(p4)*c(p5)+a3*m6*R6z*s(p5)+J6zz*c(p4)
  +J6yy*c(p5)*c(p6)*s(p4)*s(p6)+J6yy*c(p4)*s2(p6)-J6xx*c(p5)*c(p6)
  *s(p4)*s(p6)+J6xx*c(p4)*c2(p6)+J5xx*c(p4)+J5zz*c(p4)+a3*m5*R5z*s(p5)
  +d4*m5*R5z*c(p4)*c(p5);

/*** D[5,3] ***/
*(d_matrix+(row*4)+2)= *(d_matrix+(row*2)+4);

/*** D[4,5] ***/
*(d_matrix+(row*3)+4)=-J6yy*c(p6)*s(p5)*s(p6)+J6xx*c(p6)*s(p5)*s(p6);

/*** D[5,4] ***/
*(d_matrix+(row*4)+3)= *(d_matrix+(row*3)+4);

/*** D[5,5] ***/
*(d_matrix+(row*4)+4)=J6zz+J6yy*s2(p6)+J6xx*c2(p6)+J5xx+J5zz;

/*** D[1,6] ***/
*(d_matrix+5)=-J6yy*c(p4)*s(p2+p3)*s(p5)+J6yy*c(p2+p3)*c(p5)-J6xx
  *c(p4)*s(p2+p3)*s(p5)+J6xx*c(p2+p3)*c(p5);

/*** D[6,1] ***/
*(d_matrix+(row*5))= *(d_matrix+5);

/*** D[2,6] ***/
```

```
*(d_matrix+row+5)=J6yy*s(p4)*s(p5)+J6xx*s(p4)*s(p5);

/*** D[6,2] ***/
*(d_matrix+(row*5)+1)= *(d_matrix+row+5);

/*** D[3,6] ***/
*(d_matrix+(row*2)+5)=J6yy*s(p4)*s(p5)+J6xx*s(p4)*s(p5);

/*** D[6,3] ***/
*(d_matrix+(row*5)+2)= *(d_matrix+(row*2)+5);

/*** D[4,6] ***/
*(d_matrix+(row*3)+5)=J6yy*c(p5)+J6xx*c(p5);

/*** D[6,4] ***/
*(d_matrix+(row*5)+3)= *(d_matrix+(row*3)+5);

/*** D[5,6] ***/
*(d_matrix+(row*4)+5)=0.0;

/*** D[6,5] ***/
*(d_matrix+(row*5)+4)= *(d_matrix+(row*4)+5);

/*** D[6,6] ***/
*(d_matrix+(row*5)+5)=J6yy+J6xx;

  return(d_matrix);
}
```

“make_c_veector.c”

```
#include <stdio.h>
#include <math.h>
#include "PUMA_SPEC_DEF.h"

double *Make_C_Vector(row1,col1,pos_angle,row2,col2,vel_angle)
int  row1,col1,row2,col2;
double *pos_angle,*vel_angle;
{
int i,j;
double *c_vector,
  p1,p2,p3,p4,p5,p6,  v1,v2,v3,v4,v5,v6,
  C112,C122,C113,C123,C133,C114,C124,C134,C144,C115,C125,
  C135,C145,C155,C116,C126,C136,C146,C156,C223,C233,C214,
  C224,C234,C244,C215,C225,C235,C245,C255,C216,C226,C236,
  C246,C256,C314,C324,C334,C344,C315,C325,C335,C345,C355,
  C316,C326,C336,C346,C356,C415,C425,C435,C445,C455,C416,
  C426,C436,C446,C456,C516,C526,C536,C546,C556;

  if(row1!=row2 || row1!=6 || col1!=col2 || col1!=1){
     printf("Dimension errors of arguments in Make_C_Vector().\n");
     exit();
  }

/* calloc() returns pointer of specified size whose contents */
/* are zero.                                                 */
  c_vector = (double *) calloc(row1*col1, sizeof(double));
  if(c_vector == NULL){
     printf("calloc() returned NULL in Make_C_Vector().\n");
     exit();
  }

 p1= *(pos_angle);   p2= *(pos_angle+1); p3= *(pos_angle+2);
 p4= *(pos_angle+3); p5= *(pos_angle+4); p6= *(pos_angle+5);

 v1= *(vel_angle);   v2= *(vel_angle+1); v3= *(vel_angle+2);
 v4= *(vel_angle+3); v5= *(vel_angle+4); v6= *(vel_angle+5);

/*** C1[1,2] ***/
C112=2*d4*m6*R6z*c(p2+p3)*c(p5)*s(p2+p3)-d4*m6*R6z*c(p4)*s(p5)
  +2*d4*m6*R6z*c2(p2+p3)*c(p4)*s(p5)-a3*d4*m6+2*a3*d4*m6*c2(p2+p3)
 +a2*d4*m6*c(2*p2+p3)+sq(d4)*m6*c(p2+p3)*s(p2+p3)+a2*m6*R6z
 *c(2*p2+p3)*c(p5)-a2*m6*R6z*c(p4)*s(2*p2+p3)*s(p5)-a2*a3*m6
 *s(2*p2+p3)-sq(a2)*m6*c(p2)*s(p2)-a3*m6*R6z*c(p5)+2*a3*m6*R6z
 *c2(p2+p3)*c(p5)-2*a3*m6*R6z*c(p2+p3)*c(p4)*s(p2+p3)*s(p5)
 -sq(a3)*m6*c(p2+p3)*s(p2+p3)+J6zz*c(p2+p3)*c2(p5)*s(p2+p3)
 -J6zz*c(p4)*c(p5)*s(p5)+2*J6zz*c2(p2+p3)*c(p4)*c(p5)*s(p5)
 -J6zz*c(p2+p3)*c2(p4)*s(p2+p3)*s2(p5)+J6yy*c(p2+p3)*s(p2+p3)
 *s2(p5)*s2(p6)-J6yy*c(p2+p3)*c2(p6)*s(p2+p3)*s2(p4)+J6yy*c(p6)
 *s(p4)*s(p5)*s(p6)-2*J6yy*c2(p2+p3)*c(p6)*s(p4)*s(p5)*s(p6)
 -2*J6yy*c(p2+p3)*c(p4)*c(p5)*c(p6)*s(p2+p3)*s(p4)*s(p6)+J6yy
 *c(p4)*c(p5)*s(p5)*s2(p6)-2*J6yy*c2(p2+p3)*c(p4)*c(p5)*s(p5)
 *s2(p6)-J6yy*c(p2+p3)*c2(p4)*c2(p5)*s(p2+p3)*s2(p6)-J6xx
 *c(p2+p3)*s(p2+p3)*s2(p4)*s2(p6)+J6xx*c(p2+p3)*c2(p6)*s(p2+p3)
 *s2(p5)-J6xx*c(p6)*s(p4)*s(p5)*s(p6)+2*J6xx*c2(p2+p3)*c(p6)
 *s(p4)*s(p5)*s(p6)+2*J6xx*c(p2+p3)*c(p4)*c(p5)*c(p6)*s(p2+p3)
 *s(p4)*s(p6)+J6xx*c(p4)*c(p5)*c2(p6)*s(p5)-2*J6xx*c2(p2+p3)
 *c(p4)*c(p5)*c2(p6)*s(p5)-J6xx*c(p2+p3)*c2(p4)*c2(p5)*c2(p6)
 *s(p2+p3)-a3*d4*m4+2*a3*d4*m4*c2(p2+p3)+a2*d4*m4*c(2*p2+p3)
 +sq(d4)*m4*c(p2+p3)*s(p2+p3)-a2*m4*R4y*c(2*p2+p3)-a2*a3*m4
 *s(2*p2+p3)-sq(a2)*m4*c(p2)*s(p2)+a3*m4*R4y-2*a3*m4*R4y
 *c2(p2+p3)-sq(a3)*m4*c(p2+p3)*s(p2+p3)-J4zz*c(p2+p3)*s(p2+p3)
 *s2(p4)+J4yy*c(p2+p3)*s(p2+p3)-2*d4*m4*R4y*c(p2+p3)*s(p2+p3)
 -J4xx*c(p2+p3)*c2(p4)*s(p2+p3)-sq(a2)*m2*c(p2)*s(p2)+J2yy
 *c(p2)*s(p2)-2*a2*m2*R2x*c(p2)*s(p2)-J2xx*c(p2)*s(p2)-J3xx
 *c(p2+p3)*s(p2+p3)+J3zz*c(p2+p3)*s(p2+p3)-sq(a2)*m3*c(p2)
 *s(p2)+a2*m3*R3z*c(2*p2+p3)-sq(a3)*m3*c(p2+p3)*s(p2+p3)-a2
 *a3*m3*s(2*p2+p3)+2*a3*m3*R3z*c2(p2+p3)-a3*m3*R3z-J5xx
 *c(p2+p3)*c2(p4)*c2(p5)*s(p2+p3)-2*J5xx*c2(p2+p3)*c(p4)*c(p5)
 *s(p5)+J5xx*c(p4)*c(p5)*s(p5)+J5xx*c(p2+p3)*s(p2+p3)*s2(p5)
 -J5yy*c(p2+p3)*s(p2+p3)*s2(p4)-J5zz*c(p2+p3)*c2(p4)*s(p2+p3)
 *s2(p5)+2*J5zz*c2(p2+p3)*c(p4)*c(p5)*s(p5)-J5zz*c(p4)*c(p5)
 *s(p5)+J5zz*c(p2+p3)*c2(p5)*s(p2+p3)-sq(a3)*m5*c(p2+p3)
 *s(p2+p3)-2*a3*m5*R5z*c(p2+p3)*c(p4)*s(p2+p3)*s(p5)+2*a3*m5
 *R5z*c2(p2+p3)*c(p5)-a3*m5*R5z*c(p5)-sq(a2)*m5*c(p2)*s(p2)
 -a2*a3*m5*s(2*p2+p3)-a2*m5*R5z*c(p4)*s(2*p2+p3)*s(p5)+a2*m5
 *R5z*c(2*p2+p3)*c(p5)+sq(d4)*m5*c(p2+p3)*s(p2+p3)+a2*d4*m5
 *c(2*p2+p3)+2*a3*d4*m5*c2(p2+p3)-a3*d4*m5+2*d4*m5*R5z
 *c2(p2+p3)*c(p4)*s(p5)-d4*m5*R5z*c(p4)*s(p5)+2*d4*m5*R5z
 *c(p2+p3)*c(p5)*s(p2+p3);

/*** C1[2,2] ***/
C122=d4*m6*R6z*s(p2+p3)*s(p4)*s(p5)+d3*d4*m6*s(p2+p3)+a2*m6
```

```
  *R6z*c(p2)*s(p4)*s(p5)+a2*d3*m6*c(p2)+a3*m6*R6z*c(p2+p3)
  *s(p4)*s(p5)+a3*d3*m6*c(p2+p3)+J6zz*c(p5)*s(p2+p3)*s(p4)
  *s(p5)+J6zz*c(p2+p3)*c(p4)*s(p4)*s2(p5)+d3*m6*R6z*c(p5)
  *s(p2+p3)+d3*m6*R6z*c(p2+p3)*c(p4)*s(p5)+J6yy*c(p4)*c(p6)
  *s(p2+p3)*s(p5)*s(p6)-J6yy*c(p2+p3)*c(p4)*c2(p6)*s(p4)
  -J6yy*c(p5)*s(p2+p3)*s(p4)*s(p5)*s2(p6)+J6yy*c(p2+p3)
  *c(p5)*c(p6)*s(p6)-2*J6yy*c(p2+p3)*c2(p4)*c(p5)*c(p6)
  *s(p6)+J6yy*c(p2+p3)*c(p4)*c2(p5)*s(p4)*s2(p6)-J6xx
  *c(p2+p3)*c(p4)*s(p4)*s2(p6)-J6xx*c(p4)*c(p6)*s(p2+p3)
  *s(p5)*s(p6)-J6xx*c(p5)*c2(p6)*s(p2+p3)*s(p4)*s(p5)-J6xx
  *c(p2+p3)*c(p5)*c(p6)*s(p6)+2*J6xx*c(p2+p3)*c2(p4)*c(p5)
  *c(p6)*s(p6)+J6xx*c(p2+p3)*c(p4)*c2(p5)*c2(p6)*s(p4)+d3*d4
  *m4*s(p2+p3)+a2*d3*m4*c(p2)+a3*d3*m4*c(p2+p3)-J4zz*c(p2+p3)
  *c(p4)*s(p4)-d3*m4*R4y*s(p2+p3)+J4xx*c(p2+p3)*c(p4)*s(p4)
  +a2*m2*R2z*c(p2)+d3*m3*R3z*s(p2+p3)+a2*d3*m3*c(p2)+a3*d3*m3
  *c(p2+p3)+J5xx*c(p2+p3)*c(p4)*c2(p5)*s(p4)-J5xx*c(p5)*s(p2+p3)
  *s(p4)*s(p5)-J5yy*c(p2+p3)*c(p4)*s(p4)+d3*m5*R5z*c(p2+p3)
  *c(p4)*s(p5)+d3*m5*R5z*c(p5)*s(p2+p3)+J5zz*c(p2+p3)*c(p4)
  *s(p4)*s2(p5)+J5zz*c(p5)*s(p2+p3)*s(p4)*s(p5)+a3*d3*m5
  *c(p2+p3)+a3*m5*R5z*c(p2+p3)*s(p4)*s(p5)+a2*d3*m5*c(p2)+a2
  *m5*R5z*c(p2)*s(p4)*s(p5)+d3*d4*m5*s(p2+p3)+d4*m5*R5z
  *s(p2+p3)*s(p4)*s(p5);

/*** C1[1,3] ***/
C113=2*d4*m6*R6z*c(p2+p3)*c(p5)*s(p2+p3)-d4*m6*R6z*c(p4)
  *s(p5)+2*d4*m6*R6z*c2(p2+p3)*c(p4)*s(p5)-a3*d4*m6+2*a3
  *d4*m6*c2(p2+p3)+a2*d4*m6*c(p2)*c(p2+p3)+sq(d4)*m6*c(p2+p3)
  *s(p2+p3)-a3*m6*R6z*c(p5)+2*a3*m6*R6z*c2(p2+p3)*c(p5)-2*a3
  *m6*R6z*c(p2+p3)*c(p4)*s(p2+p3)*s(p5)-sq(a3)*m6*c(p2+p3)
  *s(p2+p3)-a2*a3*m6*c(p2)*s(p2+p3)+J6zz*c(p2+p3)*c2(p5)
  *s(p2+p3)-J6zz*c(p4)*c(p5)*s(p5)+2*J6zz*c2(p2+p3)*c(p4)
  *c(p5)*s(p5)-J6zz*c(p2+p3)*c2(p4)*s(p2+p3)*s2(p5)+a2*m6
  *R6z*c(p2)*c(p2+p3)*c(p5)-a2*m6*R6z*c(p2)*c(p4)*s(p2+p3)
  *s(p5)+J6yy*c(p2+p3)*s(p2+p3)*s2(p5)*s2(p6)-J6yy*c(p2+p3)
  *c2(p6)*s(p2+p3)*s2(p4)+J6yy*c(p6)*s(p4)*s(p5)*s(p6)-2*J6yy
  *c2(p2+p3)*c(p6)*s(p4)*s(p5)*s(p6)-2*J6yy*c(p2+p3)*c(p4)
  *c(p5)*c(p6)*s(p2+p3)*s(p4)*s(p6)+J6yy*c(p4)*c(p5)*s(p5)
  *s2(p6)-2*J6yy*c2(p2+p3)*c(p4)*c(p5)*s(p5)*s2(p6)-J6yy
  *c(p2+p3)*c2(p4)*c2(p5)*s(p2+p3)*s2(p6)-J6xx*c(p2+p3)
  *s(p2+p3)*s2(p4)*s2(p6)+J6xx*c(p2+p3)*c2(p6)*s(p2+p3)
  *s2(p5)-J6xx*c(p6)*s(p4)*s(p5)*s(p6)+2*J6xx*c2(p2+p3)
  *c(p6)*s(p4)*s(p5)*s(p6)+2*J6xx*c(p2+p3)*c(p4)*c(p5)
  *c(p6)*s(p2+p3)*s(p4)*s(p6)+J6xx*c(p4)*c(p5)*c2(p6)
  *s(p5)-2*J6xx*c2(p2+p3)*c(p4)*c(p5)*c2(p6)*s(p5)-J6xx
  *c(p2+p3)*c2(p4)*c2(p5)*c2(p6)*s(p2+p3)-a3*d4*m4+2*a3
  *d4*m4*c2(p2+p3)+a2*d4*m4*c(p2)*c(p2+p3)+sq(d4)*m4*c(p2+p3)
  *s(p2+p3)+a3*m4*R4y-2*a3*m4*R4y*c2(p2+p3)-sq(a3)*m4
  *c(p2+p3)*s(p2+p3)-a2*a3*m4*c(p2)*s(p2+p3)-J4zz*c(p2+p3)
  *s(p2+p3)*s2(p4)+J4yy*c(p2+p3)*s(p2+p3)-a2*m4*R4y*c(p2)
  *c(p2+p3)-2*d4*m4*R4y*c(p2+p3)*s(p2+p3)-J4xx*c(p2+p3)
  *c2(p4)*s(p2+p3)-J3xx*c(p2+p3)*s(p2+p3)+a2*m3*R3z*c(p2)
  *c(p2+p3)+J3zz*c(p2+p3)*s(p2+p3)-sq(a3)*m3*c(p2+p3)*s(p2+p3)
  -a2*a3*m3*c(p2)*s(p2+p3)+2*a3*m3*R3z*c2(p2+p3)-a3*m3*R3z
  -J5xx*c(p2+p3)*c2(p4)*c2(p5)*s(p2+p3)-2*J5xx*c2(p2+p3)*c(p4)
  *c(p5)*s(p5)+J5xx*c(p4)*c(p5)*s(p5)+J5xx*c(p2+p3)*s(p2+p3)
  *s2(p5)-J5yy*c(p2+p3)*s(p2+p3)*s2(p4)-a2*m5*R5z*c(p2)*c(p4)
  *s(p2+p3)*s(p5)+a2*m5*R5z*c(p2)*c(p2+p3)*c(p5)-J5zz*c(p2+p3)
  *c2(p4)*s(p2+p3)*s2(p5)+2*J5zz*c2(p2+p3)*c(p4)*c(p5)*s(p5)
  -J5zz*c(p4)*c(p5)*s(p5)+J5zz*c(p2+p3)*c2(p5)*s(p2+p3)-a2*a3
  *m5*c(p2)*s(p2+p3)-sq(a3)*m5*c(p2+p3)*s(p2+p3)-2*a3*m5*R5z
  *c(p2+p3)*c(p4)*s(p2+p3)*s(p5)+2*a3*m5*R5z*c2(p2+p3)*c(p5)
  -a3*m5*R5z*c(p5)+sq(d4)*m5*c(p2+p3)*s(p2+p3)+a2*d4*m5*c(p2)
  *c(p2+p3)+2*a3*d4*m5*c2(p2+p3)-a3*d4*m5+2*d4*m5*R5z*c2(p2+p3)
  *c(p4)*s(p5)-d4*m5*R5z*c(p4)*s(p5)+2*d4*m5*R5z*c(p2+p3)*c(p5)
  *s(p2+p3);

/*** C1[2,3] ***/
C123=d4*m6*R6z*s(p2+p3)*s(p4)*s(p5)+d3*d4*m6*s(p2+p3)+a3*m6*R6z
  *c(p2+p3)*s(p4)*s(p5)+a3*d3*m6*c(p2+p3)+J6zz*c(p5)*s(p2+p3)
  *s(p4)*s(p5)+J6zz*c(p2+p3)*c(p4)*s(p4)*s2(p5)+d3*m6*R6z*c(p5)
  *s(p2+p3)+d3*m6*R6z*c(p2+p3)*c(p4)*s(p5)+J6yy*c(p4)*c(p6)
  *s(p2+p3)*s(p5)*s(p6)-J6yy*c(p2+p3)*c(p4)*c2(p6)*s(p4)-J6yy
  *c(p5)*s(p2+p3)*s(p4)*s(p5)*s2(p6)+J6yy*c(p2+p3)*c(p5)*c(p6)
  *s(p6)-2*J6yy*c(p2+p3)*c2(p4)*c(p5)*c(p6)*s(p6)+J6yy*c(p2+p3)
  *c(p4)*c2(p5)*s(p4)*s2(p6)-J6xx*c(p2+p3)*c(p4)*s(p4)*s2(p6)
  -J6xx*c(p4)*c(p6)*s(p2+p3)*s(p5)*s(p6)-J6xx*c(p5)*c2(p6)
  *s(p2+p3)*s(p4)*s(p5)-J6xx*c(p2+p3)*c(p5)*c(p6)*s(p6)+2*J6xx
  *c(p2+p3)*c2(p4)*c(p5)*c(p6)*s(p6)+J6xx*c(p2+p3)*c(p4)*c2(p5)
  *c2(p6)*s(p4)+d3*d4*m4*s(p2+p3)+a3*d3*m4*c(p2+p3)-J4zz*c(p2+p3)
  *c(p4)*s(p4)-d3*m4*R4y*s(p2+p3)+J4xx*c(p2+p3)*c(p4)*s(p4)+d3
  *m3*R3z*s(p2+p3)+a3*d3*m3*c(p2+p3)+J5xx*c(p2+p3)*c(p4)*c2(p5)
  *s(p4)-J5xx*c(p5)*s(p2+p3)*s(p4)*s(p5)-J5yy*c(p2+p3)*c(p4)
  *s(p4)+d3*m5*R5z*c(p2+p3)*c(p4)*s(p5)+d3*m5*R5z*c(p5)*s(p2+p3)
  +J5zz*c(p2+p3)*c(p4)*s(p4)*s2(p5)+J5zz*c(p5)*s(p2+p3)*s(p4)
  *s(p5)+a3*d3*m5*c(p2+p3)+a3*m5*R5z*c(p2+p3)*s(p4)*s(p5)+d3*d4
  *m5*s(p2+p3)+d4*m5*R5z*s(p2+p3)*s(p4)*s(p5);

/*** C1[3,3] ***/
C133=d4*m6*R6z*s(p2+p3)*s(p4)*s(p5)+d3*d4*m6*s(p2+p3)+a3*m6*R6z
  *c(p2+p3)*s(p4)*s(p5)+a3*d3*m6*c(p2+p3)+J6zz*c(p5)*s(p2+p3)
  *s(p4)*s(p5)+J6zz*c(p2+p3)*c(p4)*s(p4)*s2(p5)+d3*m6*R6z*c(p5)
  *s(p2+p3)+d3*m6*R6z*c(p2+p3)*c(p4)*s(p5)+J6yy*c(p4)*c(p6)
  *s(p2+p3)*s(p5)*s(p6)-J6yy*c(p2+p3)*c(p4)*c2(p6)*s(p4)-J6yy
  *c(p5)*s(p2+p3)*s(p4)*s(p5)*s2(p6)+J6yy*c(p2+p3)*c(p5)*c(p6)
  *s(p6)-2*J6yy*c(p2+p3)*c2(p4)*c(p5)*c(p6)*s(p6)+J6yy*c(p2+p3)
  *c(p4)*c2(p5)*s(p4)*s2(p6)-J6xx*c(p2+p3)*c(p4)*s(p4)*s2(p6)
  -J6xx*c(p4)*c(p6)*s(p2+p3)*s(p5)*s(p6)-J6xx*c(p5)*c2(p6)*s(p2+p3)
  *s(p4)*s(p5)-J6xx*c(p2+p3)*c(p5)*c(p6)*s(p6)+2*J6xx*c(p2+p3)
  *c2(p4)*c(p5)*c(p6)*s(p6)+J6xx*c(p2+p3)*c(p4)*c2(p5)*c2(p6)
  *s(p4)+d3*d4*m4*s(p2+p3)+a3*d3*m4*c(p2+p3)-J4zz*c(p2+p3)*c(p4)
  *s(p4)-d3*m4*R4y*s(p2+p3)+J4xx*c(p2+p3)*c(p4)*s(p4)+d3*m3*R3z
  *s(p2+p3)+a3*d3*m3*c(p2+p3)+J5xx*c(p2+p3)*c(p4)*c2(p5)*s(p4)
  -J5xx*c(p5)*s(p2+p3)*s(p4)*s(p5)-J5yy*c(p2+p3)*c(p4)*s(p4)
  +d3*m5*R5z*c(p2+p3)*c(p4)*s(p5)+d3*m5*R5z*c(p5)*s(p2+p3)+J5zz
  *c(p2+p3)*c(p4)*s(p4)*s2(p5)+J5zz*c(p5)*s(p2+p3)*s(p4)*s(p5)
  +a3*d3*m5*c(p2+p3)+a3*m5*R5z*c(p2+p3)*s(p4)*s(p5)+d3*d4*m5
  *s(p2+p3)+d4*m5*R5z*s(p2+p3)*s(p4)*s(p5);

/*** C1[1,4] ***/
C114=-J6zz*c(p2+p3)*c(p5)*s(p2+p3)*s(p4)*s(p5)+J6zz*c(p4)
  *s2(p2+p3)*s(p4)*s2(p5)-a3*m6*R6z*c2(p2+p3)*s(p4)*s(p5)-a2*m6
  *R6z*c(p2)*c(p2+p3)*s(p4)*s(p5)-d4*m6*R6z*c(p2+p3)*s(p2+p3)
  *s(p4)*s(p5)+d3*m6*R6z*c(p4)*s(p5)-J6yy*c(p2+p3)*c(p4)*c(p6)
  *s(p2+p3)*s(p5)*s(p6)-J6yy*c(p4)*c2(p6)*s2(p2+p3)*s(p4)+J6yy
  *c(p2+p3)*c(p5)*s(p2+p3)*s(p4)*s(p5)*s2(p6)+J6yy*c(p5)*c(p6)
  *s2(p2+p3)*s(p6)-2*J6yy*c2(p4)*c(p5)*c(p6)*s2(p2+p3)*s(p6)
  +J6yy*c(p4)*c2(p5)*s2(p2+p3)*s(p4)*s2(p6)-J6xx*c(p4)*s2(p2+p3)
  *s(p4)*s2(p6)+J6xx*c(p2+p3)*c(p4)*c(p6)*s(p2+p3)*s(p5)*s(p6)
  +J6xx*c(p2+p3)*c(p5)*c2(p6)*s(p2+p3)*s(p4)*s(p5)-J6xx*c(p5)
  *c(p6)*s2(p2+p3)*s(p6)+2*J6xx*c2(p4)*c(p5)*c(p6)*s2(p2+p3)
  *s(p6)+J6xx*c(p4)*c2(p5)*c2(p6)*s2(p2+p3)*s(p4)-J4zz*c(p4)
  *s2(p2+p3)*s(p4)+J4xx*c(p4)*s2(p2+p3)*s(p4)+J5xx*c(p4)*c2(p5)
  *s2(p2+p3)*s(p4)+J5xx*c(p2+p3)*c(p5)*s(p2+p3)*s(p4)*s(p5)-J5yy
  *c(p4)*s2(p2+p3)*s(p4)+d3*m5*R5z*c(p4)*s(p5)-d4*m5*R5z*c(p2+p3)
  *s(p2+p3)*s(p4)*s(p5)-a2*m5*R5z*c(p2)*c(p2+p3)*s(p4)*s(p5)
  -a3*m5*R5z*c2(p2+p3)*s(p4)*s(p5)+J5zz*c(p4)*s2(p2+p3)*s(p4)
  *s2(p5)-J5zz*c(p2+p3)*c(p5)*s(p2+p3)*s(p4)*s(p5);

/*** C1[2,4] ***/
C124=-J6zz*s(p2+p3)*s2(p4)*s2(p5)-d3*m6*R6z*s(p2+p3)*s(p4)*s(p5)
  -J6yy*c2(p4)*c2(p6)*s(p2+p3)+2*J6yy*c(p4)*c(p5)*c(p6)*s(p2+p3)
  *s(p4)*s(p6)-J6yy*c2(p5)*s(p2+p3)*s2(p4)*s2(p6)-J6xx*c2(p4)
  *s(p2+p3)*s2(p6)-2*J6xx*c(p4)*c(p5)*c(p6)*s(p2+p3)*s(p4)*s(p6)
  -J6xx*c2(p5)*c2(p6)*s(p2+p3)*s2(p4)-J4zz*c2(p4)*s(p2+p3)-J4xx
  *s(p2+p3)*s2(p4)-J5xx*c2(p5)*s(p2+p3)*s2(p4)-J5yy*c2(p4)*s(p2+p3)
  -d3*m5*R5z*s(p2+p3)*s(p4)*s(p5)-J5zz*s(p2+p3)*s2(p4)*s2(p5);

/*** C1[3,4] ***/
C134=-J6zz*s(p2+p3)*s2(p4)*s2(p5)-d3*m6*R6z*s(p2+p3)*s(p4)*s(p5)
  -J6yy*c2(p4)*c2(p6)*s(p2+p3)+2*J6yy*c(p4)*c(p5)*c(p6)*s(p2+p3)
  *s(p4)*s(p6)-J6yy*c2(p5)*s(p2+p3)*s2(p4)*s2(p6)-J6xx*c2(p4)
  *s(p2+p3)*s2(p6)-2*J6xx*c(p4)*c(p5)*c(p6)*s(p2+p3)*s(p4)*s(p6)
  -J6xx*c2(p5)*c2(p6)*s(p2+p3)*s2(p4)-J4zz*c2(p4)*s(p2+p3)-J4xx
  *s(p2+p3)*s2(p4)-J5xx*c2(p5)*s(p2+p3)*s2(p4)-J5yy*c2(p4)*s(p2+p3)
  -d3*m5*R5z*s(p2+p3)*s(p4)*s(p5)-J5zz*s(p2+p3)*s2(p4)*s2(p5);

/*** C1[4,4] ***/
C144=-J6zz*c(p5)*s(p2+p3)*s(p4)*s(p5)-a3*m6*R6z*c(p2+p3)*s(p4)
  *s(p5)-a2*m6*R6z*c(p2)*s(p4)*s(p5)-d4*m6*R6z*s(p2+p3)*s(p4)
  *s(p5)+d3*m6*R6z*c(p2+p3)*c(p4)*s(p5)-J6yy*c(p4)*c(p6)*s(p2+p3)
  *s(p5)*s(p6)+J6yy*c(p5)*s(p2+p3)*s(p4)*s(p5)*s2(p6)+J6xx*c(p4)
  *c(p6)*s(p2+p3)*s(p5)*s(p6)+J6xx*c(p5)*c2(p6)*s(p2+p3)*s(p4)
  *s(p5)+J5xx*c(p5)*s(p2+p3)*s(p4)*s(p5)+d3*m5*R5z*c(p2+p3)*c(p4)
  *s(p5)-d4*m5*R5z*s(p2+p3)*s(p4)*s(p5)-a2*m5*R5z*c(p2)*s(p4)
  *s(p5)-a3*m5*R5z*c(p2+p3)*s(p4)*s(p5)-J5zz*c(p5)*s(p2+p3)*s(p4)
  *s(p5);

/*** C1[1,5] ***/
C115=J6zz*c(p5)*s2(p4)*s(p5)-J6zz*c(p5)*s2(p2+p3)*s(p5)-J6zz*c(p2+p3)
  *c(p4)*s(p2+p3)+2*J6zz*c(p2+p3)*c(p4)*c2(p5)*s(p2+p3)+J6zz
  *c2(p2+p3)*c2(p4)*c(p5)*s(p5)-a3*m6*R6z*c(p2+p3)*s(p2+p3)*s(p5)
  +a3*m6*R6z*c2(p2+p3)*c(p4)*c(p5)-a2*m6*R6z*c(p2)*s(p2+p3)*s(p5)
  +a2*m6*R6z*c(p2)*c(p2+p3)*c(p4)*c(p5)-d4*m6*R6z*s2(p2+p3)*s(p5)
  +d4*m6*R6z*c(p2+p3)*c(p4)*c(p5)*s(p2+p3)+d3*m6*R6z*c(p5)*s(p4)
  -J6yy*c(p5)*s2(p4)*s(p5)*s2(p6)+J6yy*c(p5)*s2(p2+p3)*s(p5)*s2(p6)
  -J6yy*c(p2+p3)*c(p5)*c(p6)*s(p2+p3)*s(p4)*s(p6)+J6yy*c(p2+p3)*c(p4)
  *s(p2+p3)*s2(p6)+J6yy*c(p4)*c(p6)*s2(p2+p3)*s(p4)*s(p5)*s(p6)
  -2*J6yy*c(p2+p3)*c(p4)*c2(p5)*s(p2+p3)*s2(p6)-J6yy*c2(p2+p3)
  *c2(p4)*c(p5)*s(p5)*s2(p6)+J6xx*c(p2+p3)*c(p5)*c(p6)*s(p2+p3)
  *s(p4)*s(p6)+J6xx*c(p5)*c2(p6)*s2(p2+p3)*s(p5)-J6xx*c(p5)*c2(p6)
  *s2(p4)*s(p5)-J6xx*c(p4)*c(p6)*s2(p2+p3)*s(p4)*s(p5)*s(p6)+J6xx
  *c(p2+p3)*c(p4)*c2(p6)*s(p2+p3)-2*J6xx*c(p2+p3)*c(p4)*c2(p5)*c2(p6)
  *s(p2+p3)-J6xx*c2(p2+p3)*c2(p4)*c(p5)*c2(p6)*s(p5)-J5xx*c2(p2+p3)
  *c2(p4)*c(p5)*s(p5)-2*J5xx*c(p2+p3)*c(p4)*c2(p5)*s(p2+p3)+J5xx
  *c(p2+p3)*c(p4)*s(p2+p3)-J5xx*c(p5)*s2(p4)*s(p5)+J5xx*c(p5)
  *s2(p2+p3)*s(p5)+d3*m5*R5z*c(p5)*s(p4)+d4*m5*R5z*c(p2+p3)*c(p4)
  *c(p5)*s(p2+p3)-d4*m5*R5z*s2(p2+p3)*s(p5)+a2*m5*R5z*c(p2)*c(p2+p3)
  *c(p4)*c(p5)-a2*m5*R5z*c(p2)*s(p2+p3)*s(p5)+a3*m5*R5z*c2(p2+p3)
  *c(p4)*c(p5)-a3*m5*R5z*c(p2+p3)*s(p2+p3)*s(p5)+J5zz*c2(p2+p3)
  *c2(p4)*c(p5)*s(p5)+2*J5zz*c(p2+p3)*c(p4)*c2(p5)*s(p2+p3)-J5zz
  *c(p2+p3)*c(p4)*s(p2+p3)-J5zz*c(p5)*s2(p2+p3)*s(p5)+J5zz*c(p5)
  *s2(p4)*s(p5);

/*** C1[2,5] ***/
C125=J6zz*c(p2+p3)*s(p4)*s2(p5)+J6zz*c(p4)*c(p5)*s(p2+p3)*s(p4)
  *s(p5)+d3*m6*R6z*c(p2+p3)*s(p5)+d3*m6*R6z*c(p4)*c(p5)*s(p2+p3)
  -J6yy*c(p2+p3)*c(p4)*c(p5)*c(p6)*s(p6)+J6yy*c2(p4)*c(p6)*s(p2+p3)
  *s(p5)*s(p6)+J6yy*c(p2+p3)*c2(p5)*s(p4)*s2(p6)-J6yy*c(p4)*c(p5)
  *s(p2+p3)*s(p4)*s(p5)*s2(p6)+J6xx*c(p2+p3)*c(p4)*c(p5)*c(p6)
  *s(p6)+J6xx*c(p2+p3)*c2(p5)*c2(p6)*s(p4)-J6xx*c2(p4)*c(p6)
  *s(p2+p3)*s(p5)*s(p6)-J6xx*c(p4)*c(p5)*c2(p6)*s(p2+p3)*s(p4)
  *s(p5)-J5xx*c(p4)*c(p5)*s(p2+p3)*s(p4)*s(p5)+J5xx*c(p2+p3)
  *c2(p5)*s(p4)+d3*m5*R5z*c(p4)*c(p5)*s(p2+p3)+d3*m5*R5z*c(p2+p3)
  *s(p5)+J5zz*c(p4)*c(p5)*s(p2+p3)*s(p4)*s(p5)+J5zz*c(p2+p3)*s(p4)
  *s2(p5);

/*** C1[3,5] ***/
C135=J6zz*c(p2+p3)*s(p4)*s2(p5)+J6zz*c(p4)*c(p5)*s(p2+p3)*s(p4)
  *s(p5)+d3*m6*R6z*c(p2+p3)*s(p5)+d3*m6*R6z*c(p4)*c(p5)*s(p2+p3)
  -J6yy*c(p2+p3)*c(p4)*c(p5)*c(p6)*s(p6)+J6yy*c2(p4)*c(p6)*s(p2+p3)
  *s(p5)*s(p6)+J6yy*c(p2+p3)*c2(p5)*s(p4)*s2(p6)-J6yy*c(p4)*c(p5)
  *s(p2+p3)*s(p4)*s(p5)*s2(p6)+J6xx*c(p2+p3)*c(p4)*c(p5)*c(p6)
  *s(p6)+J6xx*c(p2+p3)*c2(p5)*c2(p6)*s(p4)-J6xx*c2(p4)*c(p6)*s(p2+p3)
  *s(p5)*s(p6)-J6xx*c(p4)*c(p5)*c2(p6)*s(p2+p3)*s(p4)*s(p5)-J5xx*c(p4)
  *c(p5)*s(p2+p3)*s(p4)*s(p5)+J5xx*c(p2+p3)*c2(p5)*s(p4)+d3*m5*R5z
  *c(p4)*c(p5)*s(p2+p3)+d3*m5*R5z*c(p2+p3)*s(p5)+J5zz*c(p4)*c(p5)
  *s(p2+p3)*s(p4)*s(p5)+J5zz*c(p2+p3)*s(p4)*s2(p5);

/*** C1[4,5] ***/
C145=J6zz*c(p4)*c2(p5)*s(p2+p3)+J6zz*c(p2+p3)*c(p5)*s(p5)+d3*m6
```

```
  *R6z*c(p2+p3)*c(p5)*s(p4)+a3*m6*R6z*c(p2+p3)*c(p4)*c(p5)+a2*m6*R6z
  *c(p2)*c(p4)*c(p5)+d4*m6*R6z*c(p4)*c(p5)*s(p2+p3)+J6yy*c(p4)
  *s(p2+p3)*s2(p5)*s2(p6)-J6yy*c(p2+p3)*c(p5)*s(p5)*s2(p6)+J6xx
  *c(p4)*c2(p6)*s(p2+p3)*s2(p5)-J6xx*c(p2+p3)*c(p5)*c2(p6)*s(p5)
  -J5xx*c(p2+p3)*c(p5)*s(p5)+J5xx*c(p4)*s(p2+p3)*s2(p5)+d4*m5*R5z
  *c(p4)*c(p5)*s(p2+p3)+a2*m5*R5z*c(p2)*c(p4)*c(p5)+a3*m5*R5z
  *c(p2+p3)*c(p4)*c(p5)+d3*m5*R5z*c(p2+p3)*c(p5)*s(p4)+J5zz*c(p2+p3)
  *c(p5)*s(p5)+J5zz*c(p4)*c2(p5)*s(p2+p3);

/*** C1[5,5] ***/
C155=d3*m6*R6z*c(p5)*s(p2+p3)+d3*m6*R6z*c(p2+p3)*c(p4)*s(p5)-a3
  *m6*R6z*c(p2+p3)*s(p4)*s(p5)-a2*m6*R6z*c(p2)*s(p4)*s(p5)-d4*m6
  *R6z*s(p2+p3)*s(p4)*s(p5)+J6yy*c(p4)*c(p6)*s(p2+p3)*s(p5)*s(p6)
  -J6yy*c(p2+p3)*c(p5)*c(p6)*s(p6)-J6xx*c(p4)*c(p6)*s(p2+p3)
  *s(p5)*s(p6)+J6xx*c(p2+p3)*c(p5)*c(p6)*s(p6)-d4*m5*R5z*s(p2+p3)
  *s(p4)*s(p5)-a2*m5*R5z*c(p2)*s(p4)*s(p5)-a3*m5*R5z*c(p2+p3)
  *s(p4)*s(p5)+d3*m5*R5z*c(p2+p3)*c(p4)*s(p5)+d3*m5*R5z*c(p5)
  *s(p2+p3);

/*** C1[1,6] ***/
C116=-J6yy*c2(p4)*c(p6)*s(p6)+J6yy*c2(p5)*c(p6)*s2(p4)*s(p6)+J6yy
  *c(p6)*s2(p2+p3)*s2(p5)*s(p6)-J6yy*c2(p2+p3)*c(p6)*s2(p4)*s(p6)
  +J6yy*c(p2+p3)*s(p2+p3)*s(p4)*s(p5)-2*J6yy*c(p2+p3)*c2(p6)
  *s(p2+p3)*s(p4)*s(p5)-2*J6yy*c(p2+p3)*c(p4)*c(p5)*c(p6)*s(p2+p3)
  *s(p5)*s(p6)+J6yy*c(p4)*c(p5)*s2(p2+p3)*s(p4)-2*J6yy*c(p4)*c(p5)
  *c2(p6)*s2(p2+p3)*s(p4)+J6yy*c2(p2+p3)*c2(p4)*c2(p5)*c(p6)*s(p6)
  +J6xx*c2(p4)*c(p6)*s(p6)+J6xx*c2(p2+p3)*c(p6)*s2(p4)*s(p6)-J6xx
  *c(p6)*s2(p2+p3)*s2(p5)*s(p6)-J6xx*c2(p5)*s2(p4)*s(p6)-J6xx
  *c(p2+p3)*s(p2+p3)*s(p4)*s(p5)+2*J6xx*c(p2+p3)*c2(p6)*s(p2+p3)
  *s(p4)*s(p5)+2*J6xx*c(p2+p3)*c(p4)*c(p5)*c(p6)*s(p2+p3)*s(p5)
  *s(p6)-J6xx*c(p4)*c(p5)*s2(p2+p3)*s(p4)+2*J6xx*c(p4)*c(p5)
  *c2(p6)*s2(p2+p3)*s(p4)-J6xx*c2(p2+p3)*c2(p4)*c2(p5)*c(p6)*s(p6);

/*** C1[2,6] ***/
C126=-J6yy*c(p2+p3)*c(p4)*c2(p6)*s(p5)+J6yy*c(p4)*c(p6)*s(p2+p3)
  *s(p4)*s(p6)-J6yy*c2(p4)*c(p5)*c2(p6)*s(p2+p3)+J6yy*c(p2+p3)
  *c(p5)*c(p6)*s(p4)*s(p5)*s(p6)-J6yy*c(p5)*s(p2+p3)*s2(p4)*s2(p6)
  +J6yy*c(p4)*c2(p5)*c(p6)*s(p2+p3)*s(p4)*s(p6)-J6xx*c(p4)*c(p6)
  *s(p2+p3)*s(p4)*s(p6)-J6xx*c(p2+p3)*c(p4)*s(p5)*s2(p6)-J6xx
  *c(p5)*c2(p6)*s(p2+p3)*s2(p4)-J6xx*c(p2+p3)*c(p5)*c(p6)*s(p4)
  *s(p5)*s(p6)-J6xx*c2(p4)*c(p5)*s(p2+p3)*s2(p6)-J6xx*c(p4)*c2(p5)
  *c(p6)*s(p2+p3)*s(p4)*s(p6);
```

```
/*** C1[3,6] ***/
C136=-J6yy*c(p2+p3)*c(p4)*c2(p6)*s(p5)+J6yy*c(p4)*c(p6)*s(p2+p3)
  *s(p4)*s(p6)+J6yy*c(p2+p3)*c(p5)*c(p6)*s(p4)*s(p5)*s(p6)-J6yy
  *c2(p4)*c(p5)*c2(p6)*s(p2+p3)-J6yy*c(p5)*s(p2+p3)*s2(p4)*s2(p6)
  +J6yy*c(p4)*c2(p5)*c(p6)*s(p2+p3)*s(p4)*s(p6)-J6xx*c(p4)*c(p6)
  *s(p2+p3)*s(p4)*s(p6)-J6xx*c(p2+p3)*c(p4)*s(p5)*s2(p6)-J6xx*c(p5)
  *c2(p6)*s(p2+p3)*s2(p4)-J6xx*c2(p4)*c(p5)*s(p2+p3)*s2(p6)-J6xx
  *c(p2+p3)*c(p5)*c(p6)*s(p4)*s(p5)*s(p6)-J6xx*c(p4)*c2(p5)*c(p6)
  *s(p2+p3)*s(p4)*s(p6);

/*** C1[4,6] ***/
C146=J6yy*s(p2+p3)*s(p4)*s(p5)*s2(p6)-J6yy*c(p4)*c(p5)*c(p6)
  *s(p2+p3)*s(p5)*s(p6)-J6yy*c(p2+p3)*c(p6)*s2(p5)*s(p6)+J6xx
  *c2(p6)*s(p2+p3)*s(p4)*s(p5)+J6xx*c(p4)*c(p5)*c(p6)*s(p2+p3)
  *s(p5)*s(p6)+J6xx*c(p2+p3)*c(p6)*s2(p5)*s(p6);

/*** C1[5,6] ***/
C156=-J6yy*c(p4)*c(p5)*c2(p6)*s(p2+p3)-J6yy*c(p2+p3)*c2(p6)*s(p5)
  +J6yy*c(p6)*s(p2+p3)*s(p4)*s(p6)-J6xx*c(p4)*c(p5)*s(p2+p3)
  *s2(p6)-J6xx*c(p6)*s(p2+p3)*s(p4)*s(p6)-J6xx*c(p2+p3)*s(p5)
  *s2(p6);

/*** C2[2,3] ***/
C223=a2*d4*m6*c(p3)-a2*a3*m6*s(p3)+a2*m6*R6z*c(p3)*c(p5)-a2*m6
  *R6z*c(p4)*s(p3)*s(p5)+a2*d4*m4*c(p3)-a2*a3*m4*s(p3)-a2*m4
  *R4y*c(p3)+a2*m3*R3z*c(p3)-a2*a3*m3*s(p3)-a2*m5*R5z*c(p4)
  *s(p3)*s(p5)+a2*m5*R5z*c(p3)*c(p5)-a2*a3*m5*s(p3)+a2*d4*m5*c(p3);

/*** C2[3,3] ***/
C233=a2*d4*m6*c(p3)-a2*a3*m6*s(p3)+a2*m6*R6z*c(p3)*c(p5)-a2*m6*R6z
  *c(p4)*s(p3)*s(p5)+a2*d4*m4*c(p3)-a2*a3*m4*s(p3)-a2*m4*R4y*c(p3)
  +a2*m3*R3z*c(p3)-a2*a3*m3*s(p3)-a2*m5*R5z*c(p4)*s(p3)*s(p5)+a2
  *m5*R5z*c(p3)*c(p5)-a2*a3*m5*s(p3)+a2*d4*m5*c(p3);

/*** C2[1,4] ***/
C214=-J6zz*c(p2+p3)*c(p4)*c(p5)*s(p5)+J6zz*c2(p4)*s(p2+p3)*s2(p5)
  +a3*m6*R6z*c(p4)*s(p2+p3)*s(p5)+a2*m6*R6z*c(p4)*s(p2)*s(p5)-d4
  *m6*R6z*c(p2+p3)*c(p4)*s(p5)+J6yy*c(p2+p3)*c(p6)*s(p4)*s(p5)
  *s(p6)+J6yy*c2(p6)*s(p2+p3)*s2(p4)+2*J6yy*c(p4)*c(p5)*c(p6)
  *s(p2+p3)*s(p4)*s(p6)+J6yy*c(p2+p3)*c(p4)*c(p5)*s(p5)*s2(p6)
  +J6yy*c2(p4)*c2(p5)*s(p2+p3)*s2(p6)+J6xx*s(p2+p3)*s2(p4)
  *s2(p6)-J6xx*c(p2+p3)*c(p6)*s(p4)*s(p5)*s(p6)+J6xx*c(p2+p3)
  *c(p4)*c(p5)*c2(p6)*s(p5)-2*J6xx*c(p4)*c(p5)*c(p6)*s(p2+p3)
  *s(p4)*s(p6)+J6xx*c2(p4)*c2(p5)*c2(p6)*s(p2+p3)+J4zz*s(p2+p3)
  *s2(p4)+J4xx*c2(p4)*s(p2+p3)+J5xx*c2(p4)*c2(p5)*s(p2+p3)+J5xx
  *c(p2+p3)*c(p4)*c(p5)*s(p5)+J5yy*s(p2+p3)*s2(p4)-d4*m5*R5z
  *c(p2+p3)*c(p4)*s(p5)+a2*m5*R5z*c(p4)*s(p2)*s(p5)+a3*m5*R5z
  *c(p4)*s(p2+p3)*s(p5)+J5zz*c2(p4)*s(p2+p3)*s2(p5)-J5zz*c(p2+p3)
  *c(p4)*c(p5)*s(p5);

/*** C2[2,4] ***/
C224=-J6zz*c(p4)*s(p4)*s2(p5)-a3*m6*R6z*s(p4)*s(p5)-a2*m6*R6z
  *c(p3)*s(p4)*s(p5)+J6yy*c(p4)*c2(p6)*s(p4)-J6yy*c(p5)*c(p6)
  *s(p6)+2*J6yy*c2(p4)*c(p5)*c(p6)*s(p6)-J6yy*c(p4)*c2(p5)*s(p4)
  *s2(p6)+J6xx*c(p4)*s(p4)*s2(p6)+J6xx*c(p5)*c(p6)*s(p6)-2*J6xx
  *c2(p4)*c(p5)*c(p6)*s(p6)-J6xx*c(p4)*c2(p5)*c2(p6)*s(p4)+J4zz
  *c(p4)*s(p4)-J4xx*c(p4)*s(p4)-J5xx*c(p4)*c2(p5)*s(p4)+J5yy
  *c(p4)*s(p4)-a2*m5*R5z*c(p3)*s(p4)*s(p5)-a3*m5*R5z*s(p4)*s(p5)
  -J5zz*c(p4)*s(p4)*s2(p5);

/*** C2[3,4] ***/
C234=-J6zz*c(p4)*s(p4)*s2(p5)-a3*m6*R6z*s(p4)*s(p5)-a2*m6*R6z
  *c(p3)*s(p4)*s(p5)+J6yy*c(p4)*c2(p6)*s(p4)-J6yy*c(p5)*c(p6)
  *s(p6)+2*J6yy*c2(p4)*c(p5)*c(p6)*s(p6)-J6yy*c(p4)*c2(p5)
  *s(p4)*s2(p6)+J6xx*c(p4)*s(p4)*s2(p6)+J6xx*c(p5)*c(p6)*s(p6)
  -2*J6xx*c2(p4)*c(p5)*c(p6)*s(p6)-J6xx*c(p4)*c2(p5)*c2(p6)
  *s(p4)+J4zz*c(p4)*s(p4)-J4xx*c(p4)*s(p4)-J5xx*c(p4)*c2(p5)
  *s(p4)+J5yy*c(p4)*s(p4)-a2*m5*R5z*c(p3)*s(p4)*s(p5)-a3*m5
  *R5z*s(p4)*s(p5)-J5zz*c(p4)*s(p4)*s2(p6);

/*** C2[4,4] ***/
C244=-J6zz*c(p4)*c(p5)*s(p5)-a2*m6*R6z*c(p4)*s(p3)*s(p5)-d4
  *m6*R6z*c(p4)*s(p5)+J6yy*c(p6)*s(p4)*s(p5)*s(p6)+J6yy*c(p4)
  *c(p5)*s(p5)*s2(p6)-J6xx*c(p6)*s(p4)*s(p5)*s(p6)+J6xx*c(p4)
  *c(p5)*c2(p6)*s(p5)+J5xx*c(p4)*c(p5)*s(p5)-d4*m5*R5z*c(p4)
  *s(p5)-a2*m5*R5z*c(p4)*s(p3)*s(p5)-J5zz*c(p4)*c(p5)*s(p5);

/*** C2[1,5] ***/
C215=-J6zz*c(p2+p3)*c2(p5)*s(p4)+J6zz*c(p4)*c(p5)*s(p2+p3)
  *s(p4)*s(p5)+a3*m6*R6z*c(p5)*s(p2+p3)*s(p4)+a2*m6*R6z*c(p5)
  *s(p2)*s(p4)-d4*m6*R6z*c(p2+p3)*c(p5)*s(p4)-J6yy*c(p2+p3)
  *s(p4)*s2(p5)*s2(p6)-J6yy*c(p6)*s(p2+p3)*s2(p4)*s(p5)*s(p6)
  -J6yy*c(p4)*c(p5)*s(p2+p3)*s(p4)*s(p5)*s2(p6)+J6xx*c(p6)
  *s(p2+p3)*s2(p4)*s(p5)*s(p6)-J6xx*c(p2+p3)*c2(p6)*s(p4)
  *s2(p5)-J6xx*c(p4)*c(p5)*c2(p6)*s(p2+p3)*s(p4)*s(p5)-J5xx
  *c(p4)*c(p5)*s(p2+p3)*s(p4)*s(p5)-J5xx*c(p2+p3)*s(p4)
  *s2(p5)-d4*m5*R5z*c(p2+p3)*c(p5)*s(p4)+a2*m5*R5z*c(p5)
  *s(p2)*s(p4)+a3*m5*R5z*c(p5)*s(p2+p3)*s(p4)+J5zz*c(p4)
  *c(p5)*s(p2+p3)*s(p4)*s(p5)-J5zz*c(p2+p3)*c2(p5)*s(p4);

/*** C2[2,5] ***/
C225=-J6zz*c(p5)*s2(p4)*s(p5)+a3*m6*R6z*c(p4)*c(p5)-a2*m6
  *R6z*s(p3)*s(p5)+a2*m6*R6z*c(p3)*c(p4)*c(p5)-d4*m6*R6z
  *s(p5)-J6yy*c(p4)*c(p6)*s(p4)*s(p5)*s(p6)+J6yy*c(p5)
  *s2(p4)*s(p5)*s2(p6)+J6xx*c(p4)*c(p6)*s(p4)*s(p5)*s(p6)
  +J6xx*c(p5)*c2(p6)*s2(p4)*s(p5)+J5xx*c(p5)*s2(p4)*s(p5)
  -d4*m5*R5z*s(p5)+a2*m5*R5z*c(p3)*c(p4)*c(p5)-a2*m5*R5z
  *s(p3)*s(p5)+a3*m5*R5z*c(p4)*c(p5)-J5zz*c(p5)*s2(p4)*s(p5);

/*** C2[3,5] ***/
C235=-J6zz*c(p5)*s2(p4)*s(p5)+a3*m6*R6z*c(p4)*c(p5)-a2*m6
  *R6z*s(p3)*s(p5)+a2*m6*R6z*c(p3)*c(p4)*c(p5)-d4*m6*R6z
  *s(p5)-J6yy*c(p4)*c(p6)*s(p4)*s(p5)*s(p6)+J6yy*c(p5)*s2(p4)
  *s(p5)*s2(p6)+J6xx*c(p4)*c(p6)*s(p4)*s(p5)*s(p6)+J6xx*c(p5)
  *c2(p6)*s2(p4)*s(p5)+J5xx*c(p5)*s2(p4)*s(p5)-d4*m5*R5z*s(p5)
  +a2*m5*R5z*c(p3)*c(p4)*c(p5)-a2*m5*R5z*s(p3)*s(p5)+a3*m5*R5z
  *c(p4)*c(p5)-J5zz*c(p5)*s2(p4)*s(p5);

/*** C2[4,5] ***/
C245=-J6zz*c2(p5)*s(p4)-a2*m6*R6z*c(p5)*s(p3)*s(p4)-d4*m6*R6z
  *c(p5)*s(p4)-J6yy*s(p4)*s2(p5)*s2(p6)-J6xx*c2(p6)*s(p4)*s2(p5)
  -J5xx*s(p4)*s2(p5)-d4*m5*R5z*c(p5)*s(p4)-a2*m5*R5z*c(p5)*s(p3)
  *s(p4)-J5zz*c2(p5)*s(p4);

/*** C2[5,5] ***/
C255=a3*m6*R6z*c(p5)+a2*m6*R6z*c(p3)*c(p5)-a2*m6*R6z*c(p4)*s(p3)
  *s(p5)-d4*m6*R6z*c(p4)*s(p5)-J6yy*c(p6)*s(p4)*s(p5)*s(p6)+J6xx
  *c(p6)*s(p4)*s(p5)*s(p6)-d4*m5*R5z*c(p4)*s(p5)-a2*m5*R5z*c(p4)
  *s(p3)*s(p5)+a2*m5*R5z*c(p3)*c(p5)+a3*m5*R5z*c(p5);

/*** C2[1,6] ***/
C216=J6yy*c(p2+p3)*c(p4)*s(p5)*s2(p6)+J6yy*c(p4)*c(p6)*s(p2+p3)
  *s(p4)*s(p6)+J6yy*c(p2+p3)*c(p5)*c(p6)*s(p4)*s(p5)*s(p6)+J6yy
  *c(p5)*c2(p6)*s(p2+p3)*s2(p4)+J6yy*c2(p4)*c(p5)*s(p2+p3)*s2(p6)
  +J6yy*c(p4)*c2(p5)*c(p6)*s(p2+p3)*s(p4)*s(p6)-J6xx*c(p4)*c(p6)
  *s(p2+p3)*s(p4)*s(p6)+J6xx*c(p2+p3)*c(p4)*c2(p6)*s(p5)+J6xx
  *c2(p4)*c(p5)*c2(p6)*s(p2+p3)+J6xx*c(p5)*s(p2+p3)*s2(p4)*s2(p6)
  -J6xx*c(p2+p3)*c(p5)*c(p6)*s(p4)*s(p5)*s(p6)-J6xx*c(p4)*c2(p5)
  *c(p6)*s(p2+p3)*s(p4)*s(p6);

/*** C2[2,6] ***/
C226=J6yy*c(p6)*s2(p5)*s(p6)-J6yy*c(p6)*s2(p4)*s(p6)-J6yy*c(p4)
  *c(p5)*s(p4)+2*J6yy*c(p4)*c(p5)*c2(p6)*s(p4)+J6yy*c2(p4)
  *c2(p5)*c(p6)*s(p6)+J6xx*c(p6)*s2(p4)*s(p6)-J6xx*c(p6)*s2(p5)
  *s(p6)+J6xx*c(p4)*c(p5)*s(p4)-2*J6xx*c(p4)*c(p5)*c2(p6)*s(p4)
  -J6xx*c2(p4)*c2(p5)*c(p6)*s(p6);

/*** C2[3,6] ***/
C236=J6yy*c(p6)*s2(p5)*s(p6)-J6yy*c(p6)*s2(p4)*s(p6)-J6yy*c(p4)
  *c(p5)*s(p4)+2*J6yy*c(p4)*c(p5)*c2(p6)*s(p4)+J6yy*c2(p4)*c2(p5)
  *c(p6)*s(p6)+J6xx*c(p6)*s2(p4)*s(p6)-J6xx*c(p6)*s2(p5)*s(p6)
  +J6xx*c(p4)*c(p5)*s(p4)-2*J6xx*c(p4)*c(p5)*c2(p6)*s(p4)-J6xx
  *c2(p4)*c2(p5)*c(p6)*s(p6);

/*** C2[4,6] ***/
C246=J6yy*c(p4)*s(p5)*s2(p6)+J6yy*c(p5)*c(p6)*s(p4)*s(p5)*s(p6)
  +J6xx*c(p4)*c2(p6)*s(p5)-J6xx*c(p5)*c(p6)*s(p4)*s(p5)*s(p6);

/*** C2[5,6] ***/
C256=J6yy*c(p5)*c2(p6)*s(p4)+J6yy*c(p4)*c(p6)*s(p6)+J6xx*c(p5)
  *s(p4)*s2(p6)-J6xx*c(p4)*c(p6)*s(p6);

/*** C3[1,4] ***/
C314=-J6zz*c(p2+p3)*c(p4)*c(p5)*s(p5)+J6zz*c2(p4)*s(p2+p3)*s2(p5)
  +a3*m6*R6z*c(p4)*s(p2+p3)*s(p5)-d4*m6*R6z*c(p2+p3)*c(p4)*s(p5)
  +J6yy*c(p2+p3)*c(p6)*s(p4)*s(p5)*s(p6)+J6yy*c2(p6)*s(p2+p3)
  *s2(p4)+2*J6yy*c(p4)*c(p5)*c(p6)*s(p2+p3)*s(p4)*s(p6)+J6yy
  *c(p2+p3)*c(p4)*c(p5)*s(p5)*s2(p6)+J6yy*c2(p4)*c2(p5)*s(p2+p3)
  *s2(p6)+J6xx*s(p2+p3)*s2(p4)*s2(p6)-J6xx*c(p2+p3)*c(p6)*s(p4)
  *s(p5)*s(p6)+J6xx*c(p2+p3)*c(p4)*c(p5)*c2(p6)*s(p5)-2*J6xx*c(p4)
  *c(p5)*c(p6)*s(p2+p3)*s(p4)*s(p6)+J6xx*c2(p4)*c2(p5)*c2(p6)
  *s(p2+p3)+J4zz*s(p2+p3)*s2(p4)+J4xx*c2(p4)*s(p2+p3)+J5xx*c2(p4)
  *c2(p5)*s(p2+p3)+J5xx*c(p2+p3)*c(p4)*c(p5)*s(p5)+J5yy*s(p2+p3)
  *s2(p4)-d4*m5*R5z*c(p2+p3)*c(p4)*s(p5)+a3*m5*R5z*c(p4)*s(p2+p3)
  *s(p5)+J5zz*c2(p4)*s(p2+p3)*s2(p5)-J5zz*c(p2+p3)*c(p4)*c(p5)*s(p5);

/*** C3[2,4] ***/
```

```
C324=-J6zz*c(p4)*s(p4)*s2(p5)-a3*m6*R6z*s(p4)*s(p5)+J6yy*c(p4)
  *c2(p6)*s(p4)-J6yy*c(p5)*c(p6)*s(p6)+2*J6yy*c2(p4)*c(p5)*c(p6)
  *s(p6)-J6yy*c(p4)*c2(p5)*s(p4)*s2(p6)+J6xx*c(p4)*s(p4)*s2(p6)
  +J6xx*c(p5)*c(p6)*s(p6)-2*J6xx*c2(p4)*c(p5)*c(p6)*s(p6)-J6xx
  *c(p4)*c2(p5)*c2(p6)*s(p4)+J4zz*c(p4)*s(p4)-J4xx*c(p4)*s(p4)
  -J5xx*c(p4)*c2(p5)*s(p4)+J5yy*c(p4)*s(p4)-a3*m5*R5z*s(p4)*s(p5)
  -J5zz*c(p4)*s(p4)*s2(p5);

/*** C3[3,4] ***/
C334=-J6zz*c(p4)*s(p4)*s2(p5)-a3*m6*R6z*s(p4)*s(p5)+J6yy*c(p4)
  *c2(p6)*s(p4)-J6yy*c(p5)*c(p6)*s(p6)+2*J6yy*c2(p4)*c(p5)*c(p6)
  *s(p6)-J6yy*c(p4)*c2(p5)*s(p4)*s2(p6)+J6xx*c(p4)*s(p4)*s2(p6)
  +J6xx*c(p5)*c(p6)*s(p6)-2*J6xx*c2(p4)*c(p5)*c(p6)*s(p6)-J6xx
  *c(p4)*c2(p5)*c2(p6)*s(p4)+J4zz*c(p4)*s(p4)-J4xx*c(p4)*s(p4)
  -J5xx*c(p4)*c2(p5)*s(p4)+J5yy*c(p4)*s(p4)-a3*m5*R5z*s(p4)*s(p5)
  -J5zz*c(p4)*s(p4)*s2(p5);

/*** C3[4,4] ***/
C344=-J6zz*c(p4)*c(p5)*s(p5)-d4*m6*R6z*c(p4)*s(p5)+J6yy*c(p6)*s(p4)
  *s(p5)*s(p6)+J6yy*c(p4)*c(p5)*s(p5)*s2(p6)-J6xx*c(p6)*s(p4)*s(p5)
  *s(p6)+J6xx*c(p4)*c(p5)*c2(p6)*s(p5)+J5xx*c(p4)*c(p5)*s(p5)-d4*m5
  *R5z*c(p4)*s(p5)-J5zz*c(p4)*c(p5)*s(p5);

/*** C3[1,5] ***/
C315=-J6zz*c(p2+p3)*c2(p5)*s(p4)+J6zz*c(p4)*c(p5)*s(p2+p3)*s(p4)
  *s(p5)+a3*m6*R6z*c(p5)*s(p2+p3)*s(p4)-d4*m6*R6z*c(p2+p3)*c(p5)
  *s(p4)-J6yy*c(p2+p3)*s(p4)*s2(p5)*s2(p6)-J6yy*c(p6)*s(p2+p3)
  *s2(p4)*s(p5)*s(p6)-J6yy*c(p4)*c(p5)*s(p2+p3)*s(p4)*s(p5)*s2(p6)
  +J6xx*c(p6)*s(p2+p3)*s2(p4)*s(p5)*s(p6)-J6xx*c(p2+p3)*c2(p6)*s(p4)
  *s2(p5)-J6xx*c(p4)*c(p5)*c2(p6)*s(p2+p3)*s(p4)*s(p5)-J5xx*c(p4)
  *c(p5)*s(p2+p3)*s(p4)*s(p5)-J5xx*c(p2+p3)*s(p4)*s2(p5)-d4*m5*R5z
  *c(p2+p3)*c(p5)*s(p4)+a3*m5*R5z*c(p5)*s(p2+p3)*s(p4)+J5zz*c(p4)
  *c(p5)*s(p2+p3)*s(p4)*s(p5)-J5zz*c(p2+p3)*c2(p5)*s(p4);

/*** C3[2,5] ***/
C325=-J6zz*c(p5)*s2(p4)*s(p5)+a3*m6*R6z*c(p4)*c(p5)-d4*m6*R6z*s(p5)
  -J6yy*c(p4)*c(p6)*s(p4)*s(p5)*s(p6)+J6yy*c(p5)*s2(p4)*s(p5)*s2(p6)
  +J6xx*c(p4)*c(p6)*s(p4)*s(p5)*s(p6)+J6xx*c(p5)*c2(p6)*s2(p4)*s(p5)
  +J5xx*c(p5)*s2(p4)*s(p5)-d4*m5*R5z*s(p5)+a3*m5*R5z*c(p4)*c(p5)
  -J5zz*c(p5)*s2(p4)*s(p5);

/*** C3[3,5] ***/
```

```
C335=-J6zz*c(p5)*s2(p4)*s(p5)+a3*m6*R6z*c(p4)*c(p5)-d4*m6*R6z*s(p5)
  -J6yy*c(p4)*c(p6)*s(p4)*s(p5)*s(p6)+J6yy*c(p5)*s2(p4)*s(p5)*s2(p6)
  +J6xx*c(p4)*c(p6)*s(p4)*s(p5)*s(p6)+J6xx*c(p5)*c2(p6)*s2(p4)*s(p5)
  +J5xx*c(p5)*s2(p4)*s(p5)-d4*m5*R5z*s(p5)+a3*m5*R5z*c(p4)*c(p5)
  -J5zz*c(p5)*s2(p4)*s(p5);

/*** C3[4,5] ***/
C345=-J6zz*c2(p5)*s(p4)-d4*m6*R6z*c(p5)*s(p4)-J6yy*s(p4)*s2(p5)*s2(p6)
  -J6xx*c2(p6)*s(p4)*s2(p5)-J5xx*s(p4)*s2(p5)-d4*m5*R5z*c(p5)*s(p4)
  -J5zz*c2(p5)*s(p4);

/*** C3[5,5] ***/
C355=a3*m6*R6z*c(p5)-d4*m6*R6z*c(p4)*s(p5)-J6yy*c(p6)*s(p4)*s(p5)*s(p6)
  +J6xx*c(p6)*s(p4)*s(p5)*s(p6)-d4*m5*R5z*c(p4)*s(p5)+a3*m5*R5z*c(p5);

/*** C3[1,6] ***/
C316=J6yy*c(p2+p3)*c(p4)*s(p5)*s2(p6)+J6yy*c(p4)*c(p6)*s(p2+p3)*s(p4)
  *s(p6)+J6yy*c(p2+p3)*c(p5)*c(p6)*s(p4)*s(p5)*s(p6)+J6yy*c(p5)*c2(p6)
  *s(p2+p3)*s2(p4)+J6yy*c2(p4)*c(p5)*s(p2+p3)*s2(p6)+J6yy*c(p4)*c2(p5)
  *c(p6)*s(p2+p3)*s(p4)*s(p6)-J6xx*c(p4)*c(p6)*s(p2+p3)*s(p4)*s(p6)
  +J6xx*c(p2+p3)*c(p4)*c2(p6)*s(p5)+J6xx*c2(p4)*c(p5)*c2(p6)*s(p2+p3)
  +J6xx*c(p5)*s(p2+p3)*s2(p4)*s2(p6)-J6xx*c(p2+p3)*c(p5)*c(p6)*s(p4)
  *s(p5)*s(p6)-J6xx*c(p4)*c2(p5)*c(p6)*s(p2+p3)*s(p4)*s(p6);

/*** C3[2,6] ***/
C326=J6yy*c(p6)*s2(p5)*s(p6)-J6yy*c(p6)*s2(p4)*s(p6)-J6yy*c(p4)*c(p5)
  *s(p4)+2*J6yy*c(p4)*c(p5)*c2(p6)*s(p4)+J6yy*c2(p4)*c2(p5)*c(p6)*s(p6)
  +J6xx*c(p6)*s2(p4)*s(p6)-J6xx*c(p6)*s2(p5)*s(p6)+J6xx*c(p4)*c(p5)
  *s(p4)-2*J6xx*c(p4)*c(p5)*c2(p6)*s(p4)-J6xx*c2(p4)*c2(p5)*c(p6)*s(p6);

/*** C3[3,6] ***/
C336=J6yy*c(p6)*s2(p5)*s(p6)-J6yy*c(p6)*s2(p4)*s(p6)-J6yy*c(p4)*c(p5)
  *s(p4)+2*J6yy*c(p4)*c(p5)*c2(p6)*s(p4)+J6yy*c2(p4)*c2(p5)*c(p6)*s(p6)
  +J6xx*c(p6)*s2(p4)*s(p6)-J6xx*c(p6)*s2(p5)*s(p6)+J6xx*c(p4)*c(p5)
  *s(p4)-2*J6xx*c(p4)*c(p5)*c2(p6)*s(p4)-J6xx*c2(p4)*c2(p5)*c(p6)*s(p6);

/*** C3[4,6] ***/
C346=J6yy*c(p4)*s(p5)*s2(p6)+J6yy*c(p5)*c(p6)*s(p4)*s(p5)*s(p6)+J6xx
  *c(p4)*c2(p6)*s(p5)-J6xx*c(p5)*c(p6)*s(p4)*s(p5)*s(p6);

/*** C3[5,6] ***/
C356=J6yy*c(p5)*c2(p6)*s(p4)+J6yy*c(p4)*c(p6)*s(p6)+J6xx*c(p5)*s(p4)
```

```
  *s2(p6)-J6xx*c(p4)*c(p6)*s(p6);

/*** C4[1,5] ***/
C415=-J6zz*c(p4)*s(p2+p3)*s2(p5)+J6zz*c(p2+p3)*c(p5)*s(p5)-J6yy*c(p5)
  *c(p6)*s(p2+p3)*s(p4)*s(p6)-J6yy*c(p4)*c2(p5)*s(p2+p3)*s2(p6)
  -J6yy*c(p2+p3)*c(p5)*s(p5)*s2(p6)+J6xx*c(p5)*c(p6)*s(p2+p3)*s(p4)
  *s(p6)-J6xx*c(p4)*c2(p5)*c2(p6)*s(p2+p3)-J6xx*c(p2+p3)*c(p5)*c2(p6)
  *s(p5)-J5xx*c(p2+p3)*c(p5)*s(p5)-J5xx*c(p4)*c2(p5)*s(p2+p3)+J5zz
  *c(p2+p3)*c(p5)*s(p5)-J5zz*c(p4)*s(p2+p3)*s2(p5);

/*** C4[2,5] ***/
C425=J6zz*s(p4)*s2(p5)-J6yy*c(p4)*c(p5)*c(p6)*s(p6)+J6yy*c2(p5)*s(p4)
  *s2(p6)+J6xx*c(p4)*c(p5)*c(p6)*s(p6)+J6xx*c2(p5)*c2(p6)*s(p4)+J5xx
  *c2(p5)*s(p4)+J5zz*s(p4)*s2(p5);

/*** C4[3,5] ***/
C435=J6zz*s(p4)*s2(p5)-J6yy*c(p4)*c(p5)*c(p6)*s(p6)+J6yy*c2(p5)*s(p4)
  *s2(p6)+J6xx*c(p4)*c(p5)*c(p6)*s(p6)+J6xx*c2(p5)*c2(p6)*s(p4)+J5xx
  *c2(p5)*s(p4)+J5zz*s(p4)*s2(p5);

/*** C4[4,5] ***/
C445=J6zz*c(p5)*s(p5)-J6yy*c(p5)*s(p5)*s2(p6)-J6xx*c(p5)*c2(p6)*s(p5)
  -J5xx*c(p5)*s(p5)+J5zz*c(p5)*s(p5);

/*** C4[5,5] ***/
C455=-J6yy*c(p5)*c(p6)*s(p6)+J6xx*c(p5)*c(p6)*s(p6);

/*** C4[1,6] ***/
C416=-J6yy*c2(p6)*s(p2+p3)*s(p4)*s(p5)-J6yy*c(p4)*c(p5)*c(p6)*s(p2+p3)
  *s(p5)*s(p6)-J6yy*c(p2+p3)*c(p6)*s2(p5)*s(p6)-J6xx*s(p2+p3)*s(p4)
  *s(p5)*s2(p6)+J6xx*c(p4)*c(p5)*c(p6)*s(p2+p3)*s(p5)*s(p6)+J6xx*c(p2+p3)
  *c(p6)*s2(p5)*s(p6);

/*** C4[2,6] ***/
C426=-J6yy*c(p4)*c2(p6)*s(p5)+J6yy*c(p5)*c(p6)*s(p4)*s(p5)*s(p6)
  -J6xx*c(p4)*s(p5)*s2(p6)-J6xx*c(p5)*c(p6)*s(p4)*s(p5)*s(p6);

/*** C4[3,6] ***/
C436=-J6yy*c(p4)*c2(p6)*s(p5)+J6yy*c(p5)*c(p6)*s(p4)*s(p5)*s(p6)
  -J6xx*c(p4)*s(p5)*s2(p6)-J6xx*c(p5)*c(p6)*s(p4)*s(p5)*s(p6);

/*** C4[4,6] ***/
```

```
C446=-J6yy*c(p6)*s2(p5)*s(p6)+J6xx*c(p6)*s2(p5)*s(p6);

/*** C4[5,6] ***/
C456=-J6yy*c2(p6)*s(p5)-J6xx*s(p5)*s2(p6);

/*** C5[1,6] ***/
C516=J6yy*c(p4)*c(p5)*s(p2+p3)*s2(p6)+J6yy*c(p6)*s(p2+p3)*s(p4)*s(p6)
  +J6yy*c(p2+p3)*s(p5)*s2(p6)+J6xx*c(p4)*c(p5)*c2(p6)*s(p2+p3)+J6xx
  *c(p2+p3)*c2(p6)*s(p5)-J6xx*c(p6)*s(p2+p3)*s(p4)*s(p6);

/*** C5[2,6] ***/
C526=-J6yy*c(p5)*s(p4)*s2(p6)+J6yy*c(p4)*c(p6)*s(p6)-J6xx*c(p5)*c2(p6)
  *s(p4)-J6xx*c(p4)*c(p6)*s(p6);

/** C5[3,6] ***/
C536=-J6yy*c(p5)*s(p4)*s2(p6)+J6yy*c(p4)*c(p6)*s(p6)-J6xx*c(p5)*c2(p6)
  *s(p4)-J6xx*c(p4)*c(p6)*s(p6);

/*** C5[4,6] ***/
C546=J6yy*s(p5)*s2(p6)+J6xx*c2(p6)*s(p5);

/*** C5[5,6] ***/
C556=J6yy*c(p6)*s(p6)-J6xx*c(p6)*s(p6);


/**** To Make (6x1) Column C_Vector ****/
*(c_vector)=C155*sq(v5)+2*C156*v5*v6+C144*sq(v4)+2*C145*v4*v5+2*C146
   *v4*v6+C133*sq(v3)+2*C134*v3*v4+2*C135*v3*v5+2*C136*v3*v6+C122
   *sq(v2)+2*C123*v2*v3+2*C124*v2*v4+2*C125*v2*v5+2*C126*v2*v6+2
   *C116*v1*v6+2*C115*v1*v5+2*C114*v1*v4+2*C113*v1*v3+2*C112*v1*v2;

*(c_vector+1)=C255*sq(v5)+2*C256*v5*v6+C244*sq(v4)+2*C245*v4*v5+2
   *C246*v4*v6+C233*sq(v3)+2*C234*v3*v4+2*C235*v3*v5+2*C236*v3*v6
   +2*C223*v2*v3+2*C224*v2*v4+2*C225*v2*v5+2*C226*v2*v6+2*C216*v1
   *v6+2*C215*v1*v5+2*C214*v1*v4-C112*sq(v1);

*(c_vector+2)=C355*sq(v5)
   +2*C356*v5*v6+C344*sq(v4)+2*C345*v4*v5+2*C346*v4*v6+2*C334*v3*v4
   +2*C335*v3*v5+2*C336*v3*v6-C223*sq(v2)+2*C324*v2*v4+2*C325*v2*v5
   +2*C326*v2*v6+2*C316*v1*v6+2*C315*v1*v5+2*C314*v1*v4-C113*sq(v1);

*(c_vector+3)=C455*sq(v5)
```

```
    +2*C456*v5*v6+2*C445*v4*v5+2*C446*v4*v6-C334*sq(v3)+2*C435*v3*v5
    +2*C436*v3*v6-C224*sq(v2)-2*C324*v2*v3+2*C425*v2*v5+2*C426*v2*v6
    +2*C416*v1*v6+2*C415*v1*v5-2*C314*v1*v3-2*C214*v1*v2-C114*sq(v1);

*(c_vector+4)=2*C556*v5*v6-C445*sq(v4)
    +2*C546*v4*v6-C335*sq(v3)-2*C435*v3*v4+2*C536*v3*v6-C225*sq(v2)
    -2*C325*v2*v3-2*C425*v2*v4+2*C526*v2*v6+2*C516*v1*v6-2*C415*v1*v4
    -2*C315*v1*v3-2*C215*v1*v2-C115*sq(v1);

*(c_vector+5)=-C556*sq(v5)-C446*sq(v4)-2*C546*v4*v5-C336*sq(v3)
    -2*C436*v3*v4-2*C536*v3*v5-C226*sq(v2)-2*C326*v2*v3-2*C426*v2*v4
    -2*C526*v2*v5-2*C516*v1*v5-2*C416*v1*v4-2*C316*v1*v3-2*C216*v1*v2
    -C116*sq(v1);

  return(c_vector);
}
```

"make_g_veector.c"

```
#include <stdio.h>
#include <math.h>
#include "PUMA_SPEC_DEF.h"

double *Make_G_Vector(row,col,pos_angle)
int row,col;
double *pos_angle;
{
 double *g_vector,p1,p2,p3,p4,p5,p6;

  if(row!=6 || col!=1){
     printf("There is dimension error in Make_G_Vector().\n");
     exit();
  }

/* calloc() returns pointer of specified size whose contents *
 * are zero.                                                 */
 g_vector = (double *) calloc(row*col, sizeof(double));
  if(g_vector == NULL){
     printf("calloc() returned NULL in Make_G_Vector().\n");
     exit();
  }

 p1= *(pos_angle);   p2= *(pos_angle+1); p3= *(pos_angle+2);
 p4= *(pos_angle+3); p5= *(pos_angle+4); p6= *(pos_angle+5);


/*** G[1] ***/
*(g_vector)=0.0;

/*** G[2] ***/
*(g_vector+1)=-GR*m6*R6z*c(p2+p3)*c(p4)*s(p5)-GR*m6*R6z*c(p5)
   *s(p2+p3)-a3*GR*m6*c(p2+p3)-a2*GR*m6*c(p2)-d4*GR*m6*s(p2+p3)
   +GR*m4*R4y*s(p2+p3)-a3*GR*m4*c(p2+p3)-a2*GR*m4*c(p2)-d4*GR
   *m4*s(p2+p3)-GR*m2*R2x*c(p2)-a2*GR*m2*c(p2)-a3*GR*m3*c(p2+p3)
   -a2*GR*m3*c(p2)-GR*m3*R3z*s(p2+p3)-d4*GR*m5*s(p2+p3)-a2*GR*m5
   *c(p2)-a3*GR*m5*c(p2+p3)-GR*m5*R5z*c(p5)*s(p2+p3)-GR*m5*R5z
   *c(p2+p3)*c(p4)*s(p5);

/*** G[3] ***/
*(g_vector+2)=-GR*m6*R6z*c(p5)*s(p2+p3)-GR*m6*R6z*c(p2+p3)*c(p4)
   *s(p5)-a3*GR*m6*c(p2+p3)-d4*GR*m6*s(p2+p3)+GR*m4*R4y*s(p2+p3)
   -a3*GR*m4*c(p2+p3)-d4*GR*m4*s(p2+p3)-a3*GR*m3*c(p2+p3)-GR*m3
   *R3z*s(p2+p3)-d4*GR*m5*s(p2+p3)-a3*GR*m5*c(p2+p3)-GR*m5*R5z
   *c(p2+p3)*c(p4)*s(p5)-GR*m5*R5z*c(p5)*s(p2+p3);

/*** G[4] ***/
*(g_vector+3)=GR*m6*R6z*s(p2+p3)*s(p4)*s(p5)+GR*m5*R5z*s(p2+p3)
  *s(p4)*s(p5);

/*** G[5] ***/
*(g_vector+4)=-GR*m6*R6z*c(p4)*c(p5)*s(p2+p3)-GR*m6*R6z*c(p2+p3)
  *s(p5)-GR*m5*R5z*c(p2+p3)*s(p5)-GR*m5*R5z*c(p4)*c(p5)*s(p2+p3);

/*** G[6] ***/
*(g_vector+5)=0.0;

  return(g_vector);
}
```

"PUMA_SPEC_DEF.h"

```
/* General #define for trigonometric calculation */
#define s(x) (sin(x))
#define c(x) (cos(x))
#define s2(x) ((sin(x))*(sin(x)))
#define c2(x) ((cos(x))*(cos(x)))
#define sq(x) ((x)*(x))

/* The parameters of the PUMA arm(Unimate 600 robot) are
   shown in below:                                    */
#define a2 43.2  /* Cm:Length of Structural Parameters */
#define a3 1.9   /* Cm */
#define d3 12.5  /* Cm */
#define d4 43.2  /* Cm */
#define m6 170.47  /* Gram:Mass of 6th Link */
#define m5 m6*2.39 /* Gram */
#define m4 m6*8.95 /* Gram */
#define m3 m6*36.3 /* Gram */
#define m2 m6*77.3 /* Gram */
#define m1 m6*33.5 /* Gram */
#define R1z 8.0 /* Cm:Center of 1st Link Mass in Z-coordinate */
#define R2x -21.6  /* Cm */
#define R2z 21.75  /* Cm */
#define R3z 21.6   /* Cm */
#define R4y 2.0    /* Cm */
#define R5z 2.0    /* Cm */
#define R6z 1.0    /* Cm */
#define J1xx 0.5*m1*(451.0+57.9-451.0) /* Pseudo-Inertia for 1st Link */
#define J1yy 0.5*m1*(451.0+57.9-451.0)
#define J1zz 0.5*m1*(451.0+451.0-57.9)
#define J2xx 0.5*m2*(1847.0+1408.0-565.7)
#define J2yy 0.5*m2*(565.7+1408.0-1847.0)
#define J2zz 0.5*m2*(565.7+1847.0-1408.0)
#define J3xx 0.5*m3*(679.1+36.0-672.8)
#define J3yy 0.5*m3*(672.8+36.0-679.1)
#define J3zz 0.5*m3*(672.8+679.1-36.0)
#define J4xx 0.5*m4*(21.1+31.6-31.6)
#define J4yy 0.5*m4*(21.1+31.6-21.1)
#define J4zz 0.5*m4*(31.6+21.1-31.6)
#define J5xx 0.5*m5*(11.2+6.9-6.9)
#define J5yy 0.5*m5*(6.9+6.9-11.2)
```

```
#define J5zz 0.5*m5*(6.9+11.2-6.9)
#define J6xx 0.5*m6*(33.8+0.911-33.8)
#define J6yy 0.5*m6*(33.8+0.911-33.8)
#define J6zz 0.5*m6*(33.8+33.8-0.911)

/* Acceleration of Gravity */
#define GR 980 /* Cm/(Sec*Sec) */
```